idnux
CATATAN BELAJAR GNU/LINUX
BELAJAR
Sistem Manajemen Paket
DEBIAN
BY SA
Digitally signed by Ali Ahmadi
Date: 2016.12.19 11:06:57 WIB
Location: ID
Ali Ahmadi

BELAJAR..............................

SISTEM MANAJEMEN PAKET

DEBIAN

Versi 0.0.2 (ALPHA 2)

Ali Ahmadi

https://idnux.wordpress.com

Belajar Sistem Manajemen Paket Debian

Ali Ahmadi

ISBN : -

Penyunting : -

Penerbit : idnux blog (https://idnux.wordpress.com)

KATA PENGANTAR

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

Segala puji bagi **Allah** *Subhanahu wa Ta'ala*, *Raab* seluruh alam semesta. Tiada *Ilah* (yang berhak diibadahi dengan sebenarnya) selain **Allah** dan tiada sekutu bagi-**Nya**. *Shalawat* serta *salam* semoga selalu tercurah kepada Nabi kita **Nabi Muhammad** *Shallallahu 'alaihi wa sallam* beserta seluruh keluarganya, sahabatnya dan seluruh pengikutnya hingga akhir zaman.

Sistem manajemen paket merupakan salah satu dari berbagai pengetahuan dasar yang sangat penting dan harus dikuasai oleh para pengguna komputer, terlepas sistem operasi apa yang dia gunakan. Namun, yang cukup memprihatinkan (terutama bagi diri penulis pribadi) bahan pembelajaran yang secara khusus membahas tentang sistem manajemen paket ini masih terbilang kurang, baik yang berupa buku cetak, buku elektronik (*e-book*) maupun artikel di berbagai situs *internet* termasuk di dalam *blog-blog* (baik yang diasuh oleh orang pribadi maupun komunitas pengguna **GNU/Linux**). Oleh karena itu, dengan adanya buku ini penulis berharap dapat membantu para pengguna sistem operasi **GNU/Linux** terutama bagi para pengguna pemula dalam mempelajari dan memahami tentang sistem manajemen paket di sistem operasi **GNU/Linux** khususnya yang menggunakan **Sistem Manajemen Paket Debian**.

Penulis ingin mengucapkan terima kasih bagi semua pihak yang telah banyak membantu dan mendukung penyusunan buku ini baik secara langsung maupun tidak tidak langsung, terutama kepada para anggota paguyuban **Belajar LibreOffice Indonesia** dan **Belajar GNU/Linux Indonesia** yang ada di jaringan layanan pesan instan **Telegram**.

Penulis memahami di dalam buku ini masih terdapat banyak kekurangan, kelemahan dan juga kesalahan, yang semua itu murni disebabkan oleh kelalaian maupun kekurangan ilmu yang dimiliki oleh penulis. Oleh karena itu, penulis sangat berharap adanya umpan balik dari rekan-rekan pembaca sekalian baik dalam bentuk saran, kritik maupun koreksi bagi penulis sehingga dapat membantu penulis dalam mengevaluasi dan memperbaiki buku ini menjadi karya yang lebih baik.

Akhir kata, penulis ingin mengucapkan terima kasih kepada para pembaca yang telah mengunduh buku ini. Semoga buku ini dapat membantu serta memberikan manfaat bagi yang mempergunakannya, terlepas dari banyaknya kekurangan yang terdapat di dalamnya.

Pulau Bangka, 14 Oktober 2016
12 Muharram 1438H

Penulis

DAFTAR ISI

1.1. Sekilas tentang Sistem Manajemen Paket

Sebenarnya apa sih sistem manajemen paket ini? Apa sebenarnya kegunaannya? Dan mengapa kita selalu dianjurkan untuk mempelajarinya saat kita sedang memulai belajar tentang sistem operasi **GNU/Linux**? Sebegitu pentingkah mempelajari sistem manajemen paket ini? Dan mungkin ada banyak pertanyaan-pertanyaan lain yang terlintas di dalam benak atau pikiran kita jika kita dihadapkan pada pembahasan tentang sistem manajemen paket ini. Namun mungkin kita juga akan mempertanyakan mengapa pembahasan tentang hal ini tidaklah terlalu banyak, baik melalui buku, artikel, *blog*, termasuk melalui forum daring komunitas pengguna **GNU/Linux**. Jika kita lihat, pembahasan yang ada masih kalah dengan pembahasan tentang desain grafis, administrasi jaringan dan peladen (*server*), permainan, pemrograman, bahkan dengan pembahasan tentang materi tingkat tinggi seperti *penetration testing*, *hacking* dan *digital forensic*. Padahal seperti yang kita tahu, pengetahuan tentang sistem manajemen paket ini merupakan salah satu pengetahuan dasar yang harus dipelajari dalam belajar ilmu komputer, terlepas sistem operasi apa yang sedang digunakan.

Secara sederhana, sistem manajemen paket ini dapat kita pahami sebagai sistem yang mengatur tentang tata cara memasang, menghapus, mengkonfigurasi serta meningkatkan aplikasi yang ada di dalam sebuah sistem operasi komputer. Melihat pengertian seperti ini, mungkin banyak di antara kita yang berpendapat bahwa belajar manajemen pekat merupakan kegiatan yang sangat tidak penting, karena merupakan hal yang sudah biasa kita lakukan setiap hari. Apalagi jika kita sebelumnya sudah terbiasa menggunakan sistem operasi **Microsoft Windows**, yang untuk melakukan pemasangan aplikasi hanya cukup dengan klik *next* dan *next*, sehingga hal ini juga akan membentuk sikap kita menjadi kurang peduli, menyepelekan, bahkan mengabaikannya. Namun jika kita melihat dalam praktik dan kenyataannya, melakukan kegiatan manajemen paket tidaklah sesederhana itu. Ternyata masih banyak juga yang merasa kesulitan dan menjadi frustrasi karenanya, terutama ketika sedang menemukan masalah atau kekeliruan (*error*). Padahal mungkin masalah itu sebenarnya bukanlah termasuk masalah yang cukup besar dan susah untuk diselesaikan.

Namun, sebelum kita membahas tentang sistem manajemen paket ini lebih lanjut, ada baiknya kita ketahui terlebih dahulu tentang paket (*package*) dan hal-hal yang berkaitan dengan sistem manajemen paket ini lainnya.

1.2. Pengertian Paket

Menurut arti secara bahasa, paket jika merujuk kepada **Kamus Besar Bahasa Indonesia** adalah barang yang dikirimkan dalam bungkusan melalui pos atau perusahaan ekspedisi. Selain itu, bisa juga berarti sejumlah barang (buku dan sebagainya) yang dibungkus menjadi satu yang dikirimkan atau dijual seca-

ra keseluruhan sebagai satu satuan. Atau jika merujuk kepada **Kamus Oxford**, paket (yang dalam bahasa inggris disebut *package*) adalah sebuah barang atau kumpulan barang yang diselubungi dengan kertas atau plastik, atau yang dibungkus dalam sebuah kotak. Sedangkan menurut istilah yang berlaku di dunia komputer, paket atau *package* adalah sekumpulan berkas perangkat lunak baik berupa program eksekutabel maupun kode sumber termasuk di dalamnya seluruh berkas dan data pelengkap seperti meta-data, berkas konfigurasi, dokumentasi, pustaka program, paket pengembangan (berkas yang dibutuhkan untuk membangun sebuah program), beserta berkas dan data pendukung lainnya yang disatukan dalam satu berkas arsip tunggal yang nantinya dapat digunakan untuk melakukan kegiatan pemasangan perangkat lunak ke dalam komputer. Paket ini merupakan salah satu metode pendistribusian perangkat lunak yang secara umum lazim digunakan. Dengan dibuat dalam bentuk paket, bentuk perangkat lunak menjadi ringkas, karena telah dikumpulkan menjadi satu atau beberapa berkas. Hal ini tentu akan mempermudah dalam pendistribusian perangkat lunak tersebut terutama jika didistribusikan melalui jaringan *internet*. Untuk mendapatkan dan mempergunakannya, pengguna hanya perlu mengunduh satu atau beberapa berkas saja.

Sebuah paket biasanya didistribusikan dalam format tertentu. Ada yang berupa berkas arsip biasa dengan format berkas yang umum (seperti berkas **tarball**, berkas **zip**), ada yang berupa berkas arsip tetapi yang menggunakan ekstensi nama berkas tertentu (seperti berkas dengan ekstensi **.deb**, **.rpm**, **.msi**) dan ada juga yang diarsipkan ke dalam sebuah berkas eksekutabel (seperti **.exe**, **.bin**, **.run**). Di dalam sebuah paket juga berisi informasi lebih lanjut, yang biasanya dikenal dengan istilah meta-data. Pengertian dari meta-data sendiri adalah data yang menyediakan informasi tentang data lainnya. Meta-data terdiri atas dua tipe yaitu:

1) meta-data struktural yaitu data yang menjelaskan tentang wadah dari data; dan
2) meta-data deskriptif yaitu data yang menjelaskan tentang data aplikasi atau konten (isi) dari data.

Tujuan utama dari meta-data ini adalah untuk memudahkan dalam penemuan informasi yang relevan terutama untuk penemuan sumber daya. Meta-data juga membantu dalam mengatur sumber daya elektronik, menyediakan identifikasi digital dan membantu dalam mendukung pengarsipan dan pemeliharaan sumber daya. Bantuan yang diberikan meta-data dalam penemuan sumber daya adalah dengan mengizinkan sumber daya untuk ditemukan menggunakan kriteria yang relevan, mengidentifikasi sumber daya, mengelompokkan sumber daya yang sama, membedakan sumber daya yang berbeda dan memberikan informasi lokasi sumber daya.

Meta-data yang terdapat di dalam sebuah paket biasanya berisi tentang:

1) nama paket;
2) ringkasan tentang paket;
3) penjelasan tentang paket;
4) daftar tentang berkas-berkas yang ada di dalam paket;
5) versi perangkat lunak yang terkandung di dalam paket, sebagaimana nomor rilis dari paket itu sendiri;
6) waktu, tempat dan siapa yang telah membangun paket;
7) arsitektur target dari paket;

8) checksum dari berkas yang ada di dalam paket;
9) lisensi dari perangkat lunak yang terkandung di dalam paket;
10) daftar paket lain yang dibutuhkan oleh paket agar dapat berjalan dengan baik;
11) dan lain sebagainya.

Meta-data yang berasal dari paket ini nantinya akan disimpan di dalam basis data milik manajer paket dan akan digunakan untuk membantu kerja manajer paket dalam mengelola paket-paket yang ada, seperti untuk mempermudah pencarian paket, menyediakan data mengenai paket yang terpasang di sistem, data tentang paket yang butuh peningkatan dan lain sebagainya.

1.3. Pemasangan, Penghapusan dan Peningkatan Perangkat Lunak

Seperti yang telah kita ketahui sebelumnya, paket perangkat lunak agar dapat digunakan dan dijalankan di dalam sebuah sistem operasi, harus dilakukan proses pemasangan atau instalasi terlebih dahulu. Kegiatan pemasangan ini pada dasarnya adalah sebuah proses penyalinan berkas ke tempat tertentu di dalam sistem berkas komputer sehingga dapat dengan mudah diakses oleh sistem operasi. Hal ini berlaku secara umum di hampir semua sistem operasi, baik itu **MS-DOS**, **Microsoft Windows**, **Apple macOS**, **Unix**, **GNU/Linux** dan lain sebagainya. Beberapa aplikasi dapat dijalankan dengan hanya menyalin berkasnya ke sembarang folder atau direktori yang ada di komputer dan kemudian langsung dapat dieksekusi. Aplikasi yang lain disediakan dalam bentuk tertentu yang tidak dimungkinkan untuk dapat dieksekusi secara langsung (seperti dalam bentuk sebuah paket) sehingga diperlukan prosedur pemasangan secara khusus. Kegiatan yang umum dilakukan selama pemasangan perangkat lunak meliputi:
1) memastikan bahwa persyaratan sistem yang diperlukan terpenuhi;
2) memeriksa versi perangkat lunak;
3) membuat berkas dan folder aplikasi;
4) menambahkan data konfigurasi seperti berkas konfigurasi, entri *registry* (**Windows**), atau mengatur variabel lingkungan tempat aplikasi dipasang (contoh mengatur hak akses dan perizinan berkas dan folder aplikasi);
5) membuat perangkat lunak yang dapat diakses dengan mudah oleh pengguna, seperti dengan membuat tautan (*link*), pintasan (*shortcut*) atau penanda (*bookmark*);
6) mengkonfigurasi komponen yang berjalan secara otomatis, seperti *daemon* atau layanan sistem operasi;
7) menampilkan halaman aktivasi produk; dan
8) memperbarui versi perangkat lunak.

Namun, jika kita menggunakan sistem operasi **Unix** maupun mirip **Unix** seperti **GNU/Linux**, maka ada beberapa hal lain yang harus diperhatikan lebih dibandingkan dengan sistem operasi lain seperti halnya **Microsoft Windows**, yaitu karakteristik dari sistem operasi keluarga **Unix** ini yang merupakan sistem operasi banyak pengguna (*multi-user*), penggunaan sistem perizinan berkas dan adanya Standar Susunan Berkas (***File Hierarchy Standard***) yang mengatur tentang di mana berkas seharusnya diletakkan sesuai dengan tipe berkas dan kegunaanya. Oleh karena itu, setiap berkas yang ada harus ditentukan siapa

pemiliknya, pengguna mana saja yang dapat mengaksesnya (baca, tulis dan eksekusi) dan ke mana masing-masing berkas akan diletakkan. Penentuan kepemilikan, perizinan dan peletakan berkas ini harus dilakukan secara tepat. Hal ini tentu membuat proses pemasangan di sistem operasi keluarga **Unix** menjadi lebih rumit.

Aplikasi **Unix** dan **GNU/Linux** utamanya didistribusikan dalam bentuk paket kode sumber. Untuk dapat memasang suatu aplikasi ke dalam sistem, kita harus mengunduh paket kode sumbernya di situs web milik pengembangnya terlebih dahulu. Seperti ketika kita ingin memasang peladen web **Apache**, maka kita harus mengunduh kode sumbernya dari http://apache.org. Biasanya berkas kode sumber ini disediakan dalam bentuk berkas arsip terkompresi seperti menggunakan format berkas **.tar.gz**. Kemudian kita ekstrak berkas arsip tersebut. Setelah diekstrak, kita konfigurasi agar mendukung opsi tertentu dan sistem yang kita gunakan. Selanjutnya kita lakukan kompilasi dan akan menghasilkan berkas program eksekutabel yang akan dapat kita jalankan di sistem operasi dengan arsitektur CPU tertentu yang kita punya. Setelah mengkompilasi kode sumber, kita masih harus memasang aplikasi tersebut dengan meletakkan semua berkasnya (program eksekutabel, dokumentasi, berkas konfigurasi dan lain sebagainya) ke tempat yang tepat di dalam harddisk (seperti untuk berkas eksekutabel harus diletakkan di direktori **/usr/bin/**, pustaka di direktori **/usr/lib/**, berkas konfigurasi di direktori **/etc** dan lain sebagainya sesuai dengan aturan Standar Susunan Berkas atau ***File Hierarchy Standard***) dan memberikan izin (*permission*) yang tepat atas semua berkas tersebut. Kita mungkin juga perlu untuk melakukan langkah lainnya seperti mempersiapkan sistem dan mengkonfigurasi aplikasi agar aplikasi dapat berjalan dengan baik dan sesuai dengan yang seharusnya. Untuk membantu menyederhanakan semua langkah ini, adanya perangkat lunak *pre-compiled* menjadi semakin lazim dalam komunitas **Unix** dan **GNU/Linux**, sehingga kita dimungkinkan untuk memperoleh salinan berkas biner eksekutabel dari aplikasi yang ingin kita pasang sesuai dengan jenis mesin CPU yang kita gunakan tanpa perlu melakukan kompilasi kode sumber yang biasanya prosesnya lumayan rumit dan membutuhkan waktu yang lumayan lama.

Ketika kita menjalankan aplikasi yang baru kita pasang, kita pasti berharap dapat menemukan sesuatu hal yang dapat mendorong kita untuk dapat menggunakannya secara terus-menerus. Namun jika tidak, hal itu akan mendorong kita untuk menghapus aplikasi tersebut. Penghapusan aplikasi (*uninstall*) berlaku kebalikan dari langkah pemasangan (instalasi). Kita harus dapat mengingat langkah apa saja yang telah kita lakukan (seperti menambah akun pengguna dan lain-lain) kemudian membatalkannya. Kita juga harus dapat mengingat berkas apa saja yang telah kita pasang dan di mana kita meletakkannya, kemudian kita akan menghapusnya secara manual. Dan jika kita menyukai aplikasi yang telah kita pasang tersebut, kita mungkin menginginkan untuk selalu meningkatkannya (*upgrade*), seperti ketika telah ditemukan celah keamanan di aplikasi tersebut yang kemudian diperbaiki pada versi yang terbaru. Jika kita menemukan ada aplikasi yang harus ditingkatkan, maka kita akan membutuhkan untuk membuat cadangan (*backup*) atas berkas konfigurasinya, kemudian kita hapus aplikasi tersebut. Langkah selanjutnya adalah memasang versi baru aplikasi, kemudian menerapkan konfigurasi hasil kustomisasi kita yang sebelumnya telah kita cadangkan pada saat pemasangan kembali versi terbaru dari aplikasi tersebut.

Semua hal ini, tentulah sangat merepotkan. Apalagi jika kita melihat di dalam satu sistem operasi yang terinstall saja mungkin ada sekitar ratusan ribu berkas serta ribuan paket perangkat lunak yang harus kita kelola. Sebagai contoh, di dalam repositori **Debian GNU/Linux 8.0** (rilis stabil terakhir) terdapat sekitar 47.445 paket. Demikian juga di dalam repositori **Ubuntu Linux 14.04** terdapat sekitar 53.857 paket dan apabila dipasang seluruhnya akan menghasilkan lebih dari ratusan ribu bahkan jutaan berkas. Oleh karena itu, mengelola seluruh berkas itu sendiri walaupun secara teori dimungkinkan, tetapi tidak secara teknis. Bahkan dalam skala yang lebih kecil, mengelola satu aplikasi juga tidak sederhana. Sebagai contoh adalah aplikasi manajer paket “**dpkg**” berisi sekitar 139 berkas yang tersebar di berbagai direktori yang berbeda. Coba kita bayangkan, kita harus mencoba mengingat semua berkas tersebut dan kita juga mencoba menghapus semuanya secara manual dalam rangka menghapus dpkg dari sistem. Semua langkah yang dibutuhkan dalam mengelola perangkat lunak ini tidaklah unik hanya berlaku di sistem **Unix** maupun **GNU/Linux** saja, semua sistem operasi memiliki prosedur yang hampir sama yang harus diikuti agar perangkat lunak dapat digunakan dalam sistem. Oleh karena itu, terdapat beberapa pendekatan yang dipakai dalam rangka membantu mempermudah kegiatan pemasangan, penghapusan dan peningkatan perangkat lunak ini, yaitu:

1) Menggunakan perkakas pengelola paket level aplikasi mandiri
   Beberapa sistem operasi, seperti **MS-DOS**, tidak secara langsung menyediakan alat atau program untuk kegiatan manajemen perangkat lunak. Pemasangan aplikasi ke dalam sistem dapat dilakukan dengan dua cara. Yang pertama adalah dengan memasang perangkat lunak secara manual dengan menggunakan perkakas penyalin berkas. Perkakas penyalin berkas ini digunakan untuk meletakkan semua berkas aplikasi ke dalam tempat yang tepat di dalam sistem. Yang kedua adalah dengan menggunakan program pemasangan (*installer*) yang dibuat oleh pengembang aplikasi dan hanya dapat menangani kegiatan pemasangan untuk aplikasi tersebut saja. Untuk melakukan kegiatan penghapusan juga dapat dilakukan dengan dua cara, yaitu dengan menghapus setiap berkas aplikasi yang terpasang secara manual, atau menggunakan perkakas penghapusan yang datang bersama dengan aplikasi tersebut. Begitupun untuk melakukan peningkatan aplikasi, juga menggunakan prosedur yang serupa.
2) Menggunakan perkakas pengelola paket bawaan (level sistem)
   Jika sebelumnya ada beberapa sistem operasi yang tidak menyediakan perkakas pengelola perangkat lunak di dalam sistemnya, maka sistem operasi yang lain menyediakan perkakas ini secara bawaan. Perkakas ini dibuat agar dapat mengelola hampir keseluruhan perangkat lunak yang terpasang di dalam sistem. Dalam melakukan kegiatannya, perkakas ini biasanya akan melacak semua berkas aplikasi yang terpasang di dalam sistem. Pengetahuan ini akan sangat berguna saat ketika kita akan menghapus suatu aplikasi dari sistem. Dengan mengetahui berkas apa saja yang berhubungan dengan suatu aplikasi, maka perkakas tersebut dapat menemukan dan menghapus semua berkas milik aplikasi saat dilakukan proses penghapusan aplikasi. Perkakas pengelola perangkat lunak ini biasanya terdiri atas dua bentuk. Bentuk yang pertama lebih memfokuskan pada kegiatan kompilasi perangkat lunak dan penyalinan berkas ke dalam sistem. Perkakas akan mengotomatisasi banyak pekerjaan dasar dalam kegiatan kompilasi perangkat lunak seperti untuk mengunduh kode sumber, mengekstraknya, mengkonfigurasi, mengkompilasi dan memasangnya. Contoh perkakas ini adalah **ports** yang

digunakan di sistem operasi **FreeBSD** dan **portage** di **Gentoo Linux**. Sedangkan bentuk yang lain, lebih memfokuskan pada pemasangan perangkat lunak menggunakan perangkat lunak *pre-compiled* dan menghindari kegiatan kompilasi.

Di lingkungan sistem operasi **GNU/Linux**, pendekatan yang banyak dipilih oleh para pengembangnya adalah dengan menggunakan perkakas pengelola paket bawaan sistem yang biasanya dikenal sebagai manajer paket atau *package manager*.

1.4. Teknik dalam Pelaksanaan Manajemen Paket

Kegiatan manajemen paket dapat dilakukan menggunakan berbagai macam teknik. Teknik-teknik tersebut yang secara umum dapat digunakan terutama di lingkungan sistem operasi **GNU/Linux** adalah:

1) Memasang Paket dalam Direktori terpisah.
   Teknik ini merupakan teknik yang paling sederhana. Setiap paket dipasang dalam direktori yang terpisah. Sebagai contoh, paket **foo-1.1** akan dipasang di dalam direktori **/usr/pkg/foo-1.1** dan dibuat juga *symbolic link* (*symlink*) atau tautan simbolis dari **/usr/pkg/foo** ke **/usr/pkg/foo-1.1**. Ketika nanti dipasang versi baru **foo-1.2**, maka versi terbaru tersebut akan dipasang di direktori **/usr/pkg/foo-1.2** dan tautan yang sebelumnya akan diganti dengan tautan ke versi yang baru. Variabel lingkungan (*Environtment Variable*) seperti **PATH**, **LD_LIBRARY_PATH**, **MANPATH**, **INFOPATH** dan **CPPFLAGS** perlu untuk diperluas sehingga dapat mengikutkan direktori **/usr/pkg/foo**.
2) Manajemen Paket dengan mode *Symlink* (*Symbolic Link*)
   Teknik ini merupakan bentuk variasi dari teknik manajemen paket yang sebelumnya. Setiap paket dipasang menggunakan skema yang serupa tetapi daripada membuat tautan ke masing-masing direktori paket, masing-masing berkas ditautkan ke susunan direktori **/usr**. Hal ini akan menghilangkan kebutuhan untuk memperluas variabel lingkungan. Meskipun tautan dapat dibuat oleh pengguna secara otomatis, banyak manajer paket yang dibuat menggunakan pendekatan ini. Beberapa di antaranya adalah **Stow**, **Epkg**, **Graft** dan **Depot**. Proses pemasangan perlu dipalsukan, sehingga paket akan berpikir bahwa dirinya dipasang di direktori **/usr** meskipun sebenarnya paket terpasang dalam susuran direktori **/usr/pkg**. Pemasangan dengan cara ini tidak dapat menggunakan cara yang biasa. Sebagai contoh ketika kita memasang **libfoo-1.1**, kita kan menggunakan perintah berikut:

   ```
   ./configure --prefix=/usr/pkg/libfoo/1.1
   make
   make install
   ```

   Pemasangan akan dapat dilakukan, tetapi paket yang bergantung pada **libfoo** tidak akan terkait dengan **libfoo** seperti yang kita harapkan. Jika kita akan mengkompilasi paket yang membutuhkan **libfoo**, kita harus menautkan ke **/usr/pkg/libfoo/1.1/lib/libfoo.so.1** bukan ke **/usr/lib/libfoo.so.1** seperti yang kita harapkan. Maka untuk mengatasinya, kita dapat menggunakan variabel **DESTDIR** untuk melakukan pemasangan palsu paket.

   ```
   ./configure --prefix=/usr
   ```

```
make
make DESTDIR=/usr/pkg/libfoo/1.1 install
```

Kebanyakan paket telah mendukung pendekatan ini, namun ada beberapa yang tidak. Untuk paket yang tidak mendukung, kita dapat memasang paket secara manual atau dapat juga memasang paket yang bermasalah ke direktori **/opt**.

3) Mempergunakan Dasar Stempel Waktu (*Timestamp Based*)
   Dalam teknik ini, berkas akan diberikan stempel waktu sebelum dilakukan pemasangan paket. Setelah dipasang, perintah find dengan opsi tertentu dapat digunakan untuk menghasilkan catatan (*log*) dari semua berkas yang dipasang setelah stempel waktu berkas telah dibuat. Meskipun skema ini memiliki keuntungan berupa kesederhanaannya, skema ini memiliki dua kelemahan. Yang pertama adalah jika saat pemasangan stempel waktu berkas yang dipasang berbeda dengan waktu yang sebenarnya, maka berkas-berkas tersebut tidak dapat ditelusuri oleh manajer paket. Selain itu, skema ini juga hanya dapat digunakan ketika ada satu paket yang dipasang dalam satu waktu. Jika ada dua paket yang dipasang dalam satu waktu (contoh dalam dua konsol yang berbeda), catatan yang dihasilkan menjadi tidak reliabel (tidak dapat diandalkan).
4) Menelusuri Catatan Instalasi
   Dalam teknik ini, perintah-perintah yang dilakukan yang terdapat dalam skrip instalasi akan dicatat. Terdapat dua macam teknik yang dapat digunakan, yaitu:
   a) Menetapkan variabel lingkungan **LD_PRELOAD** agar mengarahkan pada pustaka tertentu agar dapat dimuat sebelum dilakukan proses pemasangan. Selama proses pemasangan, pustaka ini akan melacak paket yang sedang dipasang dengan menautkan dirinya pada berbagai berkas eksekutabel seperti **mv**, **cp**, **install** dan melacak ***system call*** yang memodifikasi sistem berkas (*filesystem*). Agar pendekatan ini dapat bekerja, semua berkas eksekutabel akan ditautkan secara dinamis tanpa bit **suid** atau **sgid**. Pemuatan pustaka mungkin dapat menyebabkan beberapa efek yang tidak diinginkan selama proses pemasangan. Oleh karena itu, disarankan agar dapat dilakukan beberapa pengujian untuk memastikan manajer paket tidak akan merusak apapun dan mencatat semua berkas yang sesuai.
   b) Menggunakan **strace** yang akan mencatat semua ***system call*** yang dibuat selama menjalankan skrip instalasi.
5) Membuat Berkas Arsip Paket
   Dalam teknik ini, pemasangan paket akan dipalsukan (*fake installation*) ke dalam susunan direktori seperti yang diatur dalam *Filesystem Hierarchy Standard*. Setelah dipasang, berkas yang dipasang (termasuk susunan direktorinya) akan dibuat sebagai arsip paket. Berkas arsip ini nantinya akan digunakan untuk memasang paket tersebut ke mesin lokal atau ke mesin yang lain. Salah satu pendekatan yang dapat digunakan untuk melakukan *fake installation* adalah dengan menggunakan **fakeroot**. Teknik ini merupakan teknik yang digunakan oleh sebagian besar sistem manajemen paket yang ada di lingkungan **GNU/Linux**, seperti **RPM**, **pacman**, **dpkg**, **pkgtool** dan lain sebagainya.
6) Pengelolaan berdasarkan kepada Pengguna

Dalam teknik ini, masing-masing paket akan dipasang ke lokasi standar sebagai pengguna yang terpisah. Berkas yang merupakan milik paket dapat dikenali dengan mudah dengan melakukan pengecekan ID pengguna.

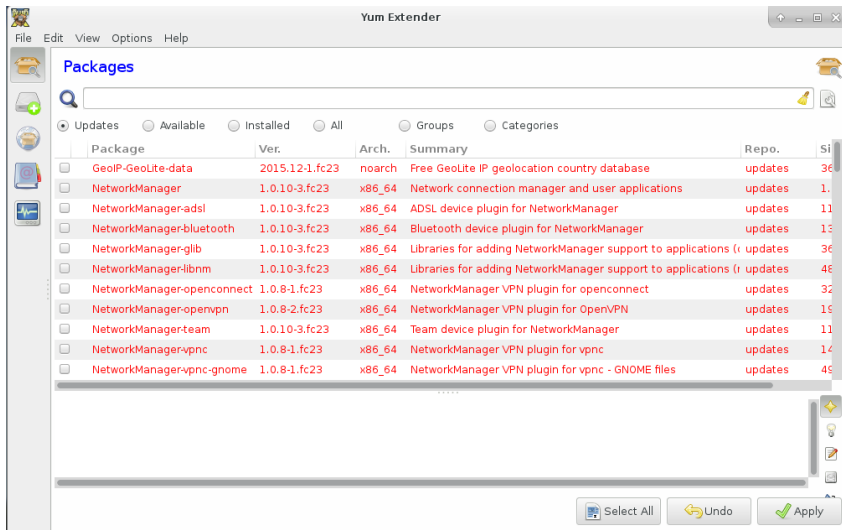
1.5. Manajer Paket

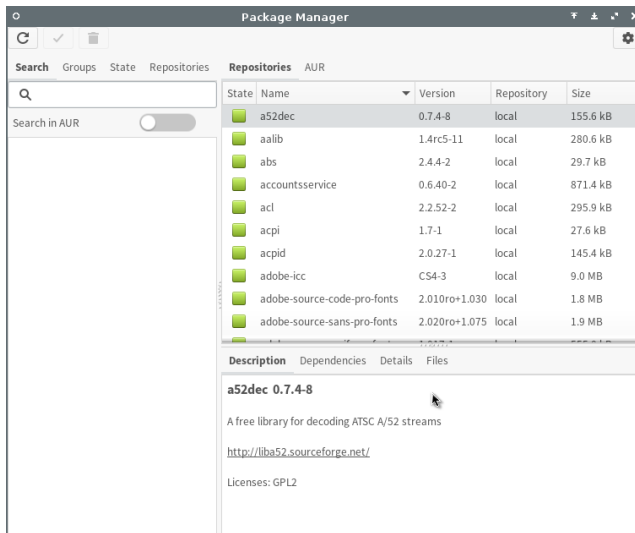
Pada awalnya, sistem operasi **GNU/Linux** tidak memiliki perkakas manajemen paket. Dahulu, jika kita ingin memasang **GNU/Linux**, kita harus melakukan *cross-compiling* atau kompilasi silang di lingkungan sistem operasi yang berbeda (sebagai contoh **Minix**), kemudian kita pasang secara manual program Linux yang telah kita kompilasi ke tempat yang tepat untuk menghasilkan sistem yang dapat bekerja dan digunakan oleh pengguna. Namun, kini sebagai sistem operasi yang sudah lumayan matang, **GNU/Linux** telah memiliki perangkat manajer paket yang membuat kegiatan pemasangan, penghapusan dan peningkatan perangkat lunak menjadi lebih mudah secara signifikan dibandingkan dengan masa awalnya. Manajer paket atau *package manager* ini menurut pengertiannya adalah alat atau program yang menyediakan metode atau cara yang konsisten dalam memasang, meningkatkan, mengkonfigurasi dan menghapus paket perangkat lunak di dalam sebuah sistem operasi komputer. Manajer paket ini merupakan program perkakas bawaan sistem operasi. Perangkat manajer paket yang digunakan dalam sistem **GNU/Linux** modern bervariasi antara distribusi yang satu dengan distribusi yang lain, namun kedua pendekatan dalam pengelolaan perangkat lunak seperti yang telah disebutkan sebelumnya dapat kita temukan dalam masing-masing variasi perangkat tersebut.

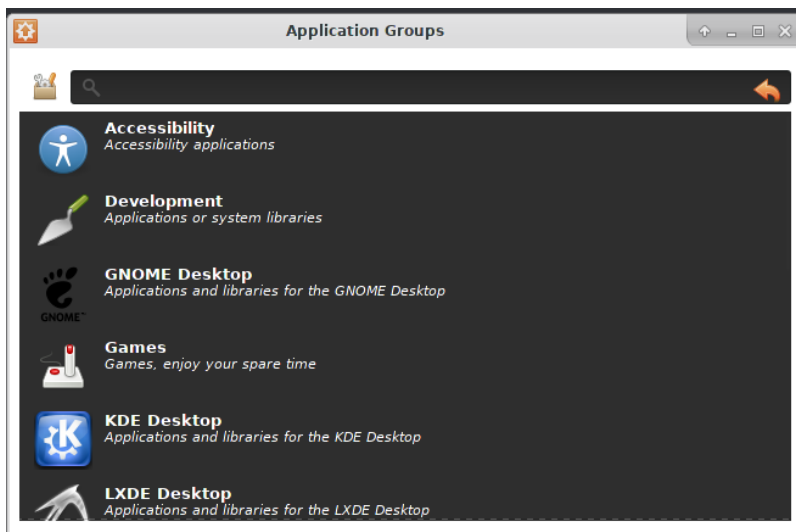
Fungsi dari perkakas manajer paket secara umum adalah sebagai berikut:

1) melakukan *query* (permintaan informasi) dan melakukan verifikasi (pembuktian atau cek kebenaran) terhadap paket;
2) melakukan pemasangan, peningkatan dan penghapusan paket-paket;
3) bekerja dengan berkas paket, contoh mengekstrak berkas paket;
4) mengelola sumber perangkat lunak (repositori) yang digunakan dan mengunduh paket dari repositori;
5) mengelompokkan paket berdasarkan fungsinya sehingga dapat mengurangi kebingungan pengguna; dan
6) mengelola dependensi untuk memastikan bahwa setiap paket dipasang bersama semua paket yang dibutuhkannya, sehingga dapat menghindari terjadinya "*dependency hell*".

Beberapa distribusi **GNU/Linux** menggunakan beberapa program manajer paket di dalam sistemnya. Hal ini dikarenakan program manajer paket utama yang digunakan hanya menyediakan fungsi dasar manajemen paket, seperti untuk pemasangan, penghapusan dan peningkatan paket. Sedangkan program manajer paket yang lain menyediakan fungsionalitas tambahan yang cukup penting seperti penanganan jaringan, pengelolaan repositori dan pengunduhan paket dari repositori, penyelesaian masalah dependensi dan lain sebagainya. Selain itu, terdapat juga program manajer paket yang dibuat untuk menyediakan antarmuka untuk program manajer paket utama, tujuannya adalah untuk memudahkan pengguna dalam melakukan kegiatan manajemen paket. Sebagai contoh adalah **RPM** sebagai perkakas manajemen paket dasar, **DNF** sebagai perkakas manajemen paket lebih lanjut dan **Yumex** (**YUM Extender**) sebagai *front-*

end dengan antarmuka grafis. Contoh lain yang serupa adalah **dpkg**, **APT** dan **synaptic**.

Gambar 1.1: Yum Extender (Yumex)

Jika suatu program manajer paket dasar telah memiliki fungsionalitas yang cukup lengkap, maka program manajer paket lainnya biasanya hanya akan memberikan sedikit tambahan fungsi saja atau hanya untuk sebagai antarmuka dari program manajer paket dasar tersebut. Perkakas manajemen paket tambahan ini mungkin tidak disediakan secara langsung oleh pengembang distribusi **GNU/Linux** sehingga harus dipasang sendiri oleh pengguna. Contoh untuk kasus yang kedua ini adalah pacman sebagai perkakas manajer paket dasar di distribusi **Arch Linux**, **yaourt** yang menyediakan fungsi penanganan untuk repositori **AUR** (*Arch User Repository*) dan **octopi** atau **pamac** sebagai *front-end* dengan antarmuka grafis.

Gambar 1.2: pamac

Selain itu, ada juga manajer paket yang mengusung konsep peladen dan klien (*server-client*). Contohnya adalah **entropy** di distribusi **Sabayon Linux** yang berperan sebagai peladen (*server*) yang berjalan di latar belakang, **equo** sebagai klien dengan antarmuka baris perintah (*command line*) dan **rigo** sebagai klien dengan antarmuka grafis.

Gambar 1.3: Rigo Application Browser

Manajer paket ini sering juga disebut sebagai manajer instalasi atau *install manager*, yang mana istilah ini dapat membantu menghilangkan kerancuan antara manajer paket dengan *installer*. Perbedaan antara keduanya tersebut meliputi:

Tabel 1.1: Perbandingan antara Manajer Paket dan Installer

No.	*Kriteria*	**Manajer Paket**	*Installer*
1	Disediakan oleh	Biasanya oleh sistem operasi	Masing-masing program komputer (aplikasi)
2	Tempat informasi instalasi	Satu basis data instalasi terpusat	Hal ini sepenuhnya bergantung pada aturan yang diterapkan oleh *installer*. Basis data bisa berupa sebuah berkas dalam folder aplikasi, atau di antara berkas atau folder milik sistem operasi. Dalam kondisi yang paling baik, *installer* akan mendaftarkan dirinya beserta daftar *uninstaller* tanpa memperlihatkan informasi tentang instalasi
3	Lingkup pengelolaan	Semua paket di dalam sistem	Hanya produk yang dibundel bersama dengan *installer*
4	Dikembangkan oleh	Satu penyedia manajer paket	Berbagai penyedia *installer*
5	Format paket	Satu atau dua format yang dapat dikenali	Mungkin ada banyak format sebanyak jumlah aplikasi yang ada
6	Kompatibilitas dengan format paket	Dapat digunakan selama didukung oleh manajer paket	*Installer* akan selalu kompatibel dengan format arsip yang digunakannya

Agar dapat bekerja, manajer paket membutuhkan berbagai macam informasi baik yang berasal dari paket (meta-data) maupun yang berasal dari luar paket seperti daftar repositori yang sedang digunakan, daftar repositori yang tersedia dan dapat digunakan oleh sistem operasi serta daftar paket yang terdapat dalam repositori tersebut. Informasi ini akan disimpan oleh manajer paket dalam bentuk basis data yang dibuat dengan format tertentu. Basis data ini biasanya berisi tentang:

1) daftar paket yang terpasang di sistem, termasuk status keadaan paket;
2) informasi yang berkaitan dengan paket, seperti nama paket, versi, gambaran tentang paket, daftar paket yang menjadi dependensi dan lain sebagainya;
3) daftar berkas milik paket yang terpasang di sistem dan letak berkas-berkas tersebut disimpan;
4) daftar paket yang tersedia di dalam repositori dan letak berkas paket yang disimpan di dalam repositori;
5) daftar repositori yang sedang digunakan oleh sistem operasi;
6) daftar repositori yang tersedia dan dapat digunakan oleh sistem operasi;
7) dan lain sebagainya, menurut kebutuhan manajer paket.

Basis data ini nantinya akan digunakan untuk membantu dalam kegiatan pengelolaan paket. Sebagai contoh, meta-data yang berasal dari paket saat dilakukan proses pemasangan akan disimpan ke dalam basis data tentang paket yang terpasang. Dengan menggunakan basis data ini manajer paket dapat lebih mu-

dah memberikan informasi tentang paket ketika diminta oleh pengguna. Contoh lainnya adalah ketika pengguna ingin memasang suatu paket ke dalam sistem, ternyata paket tersebut memiliki ketergantungan (dependensi) terhadap beberapa paket lain. Maka dengan informasi yang diperoleh dari basis data, manajer paket dapat memberitahukan kepada pengguna paket apa saya yang harus dipasang serta manajer paket jika memungkinkan dapat membantu untuk mengunduh berkas paket yang butuhkan dari repositori dan kemudian memasangnya. Begitu juga ketika pengguna ingin menghapus paket tersebut dari sistem, manajer paket dapat memanfaatkan informasi yang berasal dari basis data untuk menemukan berkas apa saja yang berhubungan dengan paket dan kemudian menghapusnya.

Paket termasuk meta-datanya harus dibuat sesuai dengan standar format yang ditetapkan oleh pengembang manajer paket. Jika tidak, maka manajer paket tidak akan dapat menggunakan informasi atau bekerja dengan paket tersebut. Standar format berkas paket ini dapat berupa format penamaan berkas paket, ekstensi nama berkas apa yang digunakan, termasuk struktur isi berkas paket secara spesifik. Sebagai contoh untuk berkas paket **RPM** menggunakan format penamaan berkas paket sebagai berikut:

```
<nama>-<nomor versi>-<rilis>.<arsitektur>.rpm
```

Sebagai contoh:

```
gedit-3.10.3-1.1.i586.rpm
```

Sedangkan format isi berkas paket biner **RPM** terdiri atas empat seksi, yaitu:

1) ***lead***, berisi tanda tangan digital (*signature*) paket (yang akan menunjukkan identitas berkas sebagai paket RPM) dan beberapa informasi tentang struktur dari paket itu sendiri;
2) ***signature*** (tanda tangan digital), yang berisi sekumpulan tanda tangan digital yang digunakan untuk menandatangani paket dengan menggunakan teknik kriptografi yang dapat digunakan untuk memastikan keutuhan dan atau keaslian berkas paket;
3) ***header*** (kepala berkas), yang berisi semua meta-data paket, meliputi nama paket, versi, arsitektur, daftar berkas, gambaran tentang paket, hubungan dengan paket yang lain dan lain sebagainya; dan
4) ***payload***, merupakan arsip berkas yang sebenarnya yang berisi semua berkas yang dibundel bersama paket. Biasanya berkas diarsipkan menggunakan format berkas **cpio** dan dikompresi menggunakan kompresi **gzip**. Beberapa versi yang lebih baru dari standar format **RPM** ini juga dapat menggunakan kompresi **bzip2**, **lzip**, **lzma**, atau **xz**. Di format **RPM** 5.0 mendukung pengarsipan menggunakan **xar**.

Tidak semua perangkat lunak yang dibuat untuk sistem operasi **GNU/Linux** didistribusikan menggunakan format paket yang sesuai dengan standar yang digunakan oleh manajer paket. Ada beberapa perangkat lunak yang hanya tersedia dalam bentuk kode sumber, beberapa lagi disediakan dalam bentuk berkas biner atau *pre-compiled* maupun *shell script* yang diarsipkan dalam sebuat berkas arsip biasa (seperti berkas **tarbal** atau **zip**) dan yang lainnya didistribusikan

menggunakan sebuah berkas biner eksekutabel atau bisa juga berupa *shell script* yang juga berfungsi sebagai program *installer*. Jika perangkat lunak hanya tersedia dalam bentuk kode sumber, maka untuk dapat memasangnya ke sistem kita harus melakukan kompilasi terlebih dahulu. Kemudian kita pasang berkas hasil kompilasi sesuai dengan petunjuk yang diberikan oleh pengembangnya.

Namun jika perangkat lunak didistribusikan dalam bentuk berkas arsip biasa yang isinya berupa berkas *pre-compiled* maupun *shell script*, langkah pertama yang harus kita lakukan adalah mengekstrak berkas arsip tersebut. Kemudian kita cari dan baca berkas yang berisi petunjuk tentang tata cara pemasangan dan penggunaan aplikasi yang bersangkutan. Petunjuk ini biasanya terdapat di dalam berkas dengan nama **README**, **INSTALL**, atau bisa juga di dalam berkas teks yang lain. Jika ternyata di dalam berkas arsip tersebut setelah kita ekstrak terdapat program eksekutabel atau skrip yang dapat digunakan untuk mengotomatisasi pelaksanaan pemasangan aplikasi, maka program atau skrip tersebut dapat kita jalankan. Namun jika tidak, maka kita harus meletakkan berkas hasil ekstraksi tadi ke tempat yang pemasangannya sesuai dengan petunjuk yang diberikan oleh pengembang. Dan jika perangkat lunak didistribusikan dalam bentuk berkas biner eksekutabel atau *shell script* yang berfungsi sebagai program *installer*, maka kita dapat langsung menjalankan berkas *installer* tersebut. Program *installer* ini biasanya lebih sering digunakan untuk pendistribusian perangkat lunak propietary terutama untuk perangkat lunak dengan lisensi komersial.

Hal yang lebih rumit mungkin akan kita temukan ketika kita ingin menghapus perangkat lunak tersebut dari sistem, karena tidak ada mekanisme atau prosedur yang sama dalam pelaksanaan penghapusan ini. Hal pertama yang harus kita lakukan adalah kita harus mencari adanya petunjuk tentang cara penghapusan aplikasi yang disarankan oleh pengembang. Selanjutnya kita baca dan ikuti petunjuk tersebut dengan benar. Penghapusan akan dapat dilakukan lebih mudah jika terdapat program atau skrip yang berfungsi sebagai *uninstaller*, atau jika terdapat mekanisme penghapusan menggunakan perintah tertentu seperti menggunakan perintah **make uninstall** (jika program di pasang langsung dari kode sumber). Namun jika tidak tersedia, maka kita harus menghapus berkas-berkas yang berhubungan dengan aplikasi tersebut secara manual.

Pemasangan paket perangkat lunak yang tidak mengikuti standar manajer paket seperti ini sebenarnya tidaklah tanpa masalah. Masalah yang paling utama adalah kemungkinan terjadinya konflik dengan paket lain atau berkas paket aplikasi mungkin akan menimpa berkas milik paket yang lain (terutama terhadap paket yang mengikuti standar manajer paket). Oleh karena itu, beberapa pengembang distribusi **GNU/Linux** sangat tidak menyarankan pemasangan paket perangkat lunak seperti ini secara langsung, kita akan disarankan untuk memasang paket di direktori yang terpisah, seperti di direktori **/usr/local** atau **/opt**. Namun, akan lebih baik jika kita dapat memaketkan sendiri perangkat lunak menjadi paket yang sesuai dengan standar. Contoh jika sistem operasi yang kita gunakan menggunakan manajer paket **dpkg**, **rpm**, atau **pkgtool** (**Slackware Linux**) sedangkan kita harus memasang aplikasi yang hanya tersedia dalam kode sumber, setelah kita lakukan kompilasi kode sumbernya kita dapat memanfaatkan perkakas **checkinstall** yang akan menghasilkan paket yang dapat dikelola oleh manajer paket tersebut. Di sistem operasi, manajer paket atau dalam kondisi yang lain mungkin harus menggunakan cara yang berbeda.

1.6. Sistem Manajemen Paket

Setelah sebelumnya kita membahas tentang paket, teori tentang pemasangan, penghapusan dan peningkatan perangkat lunak serta tentang manajer paket, maka kini kita akan membahas tentang sistem manajemen paket. Jika merujuk pada Wikipedia, sistem manajemen paket ini dijelaskan sebagai sekumpulan perangkat lunak yang mengotomatisasi proses pemasangan, peningkatan, konfigurasi dan penghapusan program komputer di dalam sistem operasi komputer dengan cara yang konsisten. Sedangkan jika merujuk pada penjelasan dalam dokumentasi milik distribusi **OpenSUSE Linux**, sistem manajemen paket dijelaskan sebagai sekumpulan alat yang menyediakan metode yang konsisten dalam memasang, meningkatkan dan menghapus perangkat lunak pada sistem operasi yang kita miliki. Namun pengertian seperti ini bagi kami terasa kurang tepat, karena tidak menggambarkan bagaimana arti dari sebuah sistem dan lebih mengarah pada arti sebagai sekumpulan alat saja.

Jika kita merujuk pada **Kamus Besar Bahasa Indonesia**, sistem mempunyai beberapa arti, yaitu:

1) perangkat unsur yg secara teratur saling berkaitan sehingga membentuk suatu totalitas. Contoh: sistem pencernaan makanan, sistem pernapasan, sistem telekomunikasi;
2) susunan yg teratur dari pandangan, teori, asas dan sebagainya. Contoh: sistem pemerintahan negara; dan
3) metode. Contoh: sistem pendidikan.

Dari pengertian seperti ini dapat kita pahami bahwa sesuatu dapat dikatakan sebagai sebuah sistem jika:

1) memiliki beberapa unsur atau komponen pembentuk;
2) unsur atau komponen tersebut tersusun secara teratur dan saling berkaitan satu sama lain;
3) membentuk sesuatu hal yang utuh dan terstruktur; dan
4) merupakan metode atau cara untuk melakukan sesuatu.

Maka sistem manajemen paket ini dapat kita artikan sebagai “sebuah sistem yang mengatur tentang kegiatan penatausahaan perangkat lunak dalam sebuah sistem operasi komputer, seperti untuk memasang, menghapus, mengkonfigurasi dan meningkatkan perangkat lunak”. Sistem manajemen paket ini terdiri atas beberapa unsur dan unsur yang utama adalah sebagai berikut:

1) paket perangkat lunak dan sistem pemaketan yang digunakan;
2) perkakas manajemen paket; dan
3) basis data manajemen paket.

Sistem manajemen paket ini memiliki beberapa keuntungan, di antaranya adalah:

1) memberikan kemudahan dalam melakukan query (permintaan informasi tentang paket termasuk versi dari suatu paket apakah sudah terpasang atau masih tersedia (dapat dipasang ke sistem);
2) memberikan kemudahan dalam melakukan penghapusan paket secara keseluruhan;
3) memberikan kemudahan dalam melakukan pelaksanaan verifikasi atas keutuhan (*integrity*) berkas paket, sehingga dapat diketahui apakah berkas paket telah rusak atau tidak;

4) memberikan kemudahan dalam melakukan peningkatan (*upgrade*) paket;
5) memberikan kemudahan dalam melihat apakah paket membutuhkan atau menyediakan sesuatu yang disediakan atau dibutuhkan oleh paket lain sehingga paket aplikasi dapat berjalan atau berfungsi dengan benar;
6) memberikan kemudahan dalam melakukan pemasangan atau penghapusan kelompok paket; dan
7) dalam beberapa kasus memberikan kemungkinan untuk dilakukan penurunan paket ke versi sebelumnya, contoh ketika di versi baru ditemukan adanya kutu (*bug*).

Selain keuntungan di atas, sistem manajemen paket ini juga memiliki sejumlah kelemahan. Salah satu kelemahan yang paling umum adalah pengguna akan dibatasi hanya dapat menggunakan paket atau versi paket tertentu yang disediakan oleh pengembang sistem operasi (di dalam repositori yang disediakan). Jika pengguna ingin menggunakan paket selain dari yang disediakan atau paket dengan versi yang berbeda, pengguna dapat membuat sendiri paket perangkat lunak tersebut. Selain itu pengguna juga dapat mencari paketnya di *website* milik pengembang atau di repositori milik pihak ketiga yang mungkin menyediakan paket tersebut.

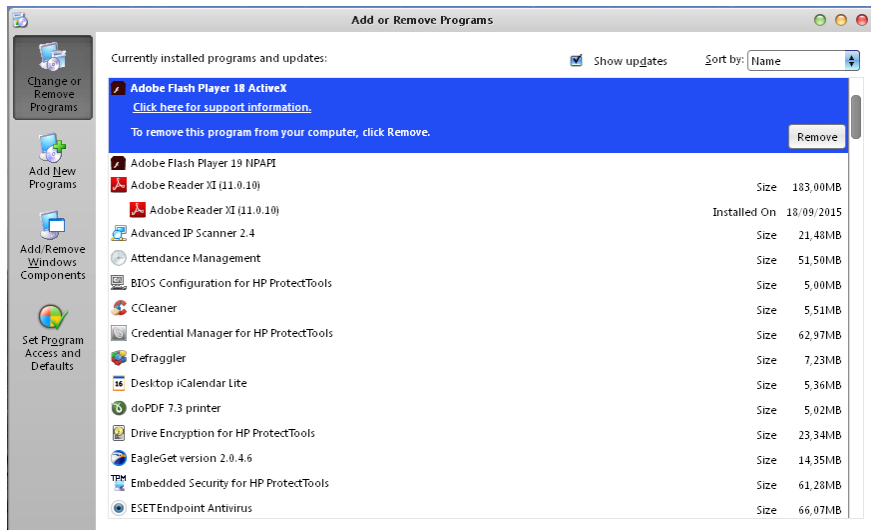
Mendiang **Ian Murdock**, yang merupakan pendiri **Proyek Debian** pernah memberikan komentar bahwa *package management is "the single biggest advancement Linux has brought to the industry"* yang artinya kurang lebih bahwa manajemen paket ini adalah salah satu kemajuan terbesar yang Linux bawa kepada industri (mungkin maksudnya adalah industri komputer dan perangkat lunak). Hal ini dikarenakan manajemen paket ini telah mengaburkan batas-batas antara sistem operasi dan aplikasi. Selain itu manajemen paket juga mempermudah dalam mendorong inovasi baru ke dalam pasar serta mendorong evolusi (perubahan atau peningkatan) sistem operasi.

Saat ini terdapat banyak macam sistem manajemen paket yang digunakan di berbagai macam sistem operasi yang ada. Beberapa di antaranya dibuat lintas platform (*cross-platform*). Khusus di lingkungan sistem operasi **GNU/Linux** dikenal berbagai macam sistem manajemen paket, di antaranya adalah:
1) **Sistem Manajemen Paket Debian**, asalnya digunakan oleh **Debian GNU/Linux** dan sekarang juga digunakan oleh distribusi **GNU/Linux** yang lain seperti **Ubuntu Linux** dan berbagai turunannya. **Sistem Manajemen Paket Debian** menggunakan format paket **.deb**;
2) **RPM *Package Manager***, dibuat oleh **Red Hat** dan digunakan oleh sejumlah distribusi **GNU/Linux** seperti **RHEL**, **Fedora**, **Mandriva**, **Mageia**, **OpenSUSE** dan lain sebagainya. **RPM** merupakan format pemaketan standar dari ***Linux Standard Base*** (LSB) dasar dari sejumlah perkakas lain seperti **apt4rpm**, **dnf**, **YUM**, **urpmi** dan **zypper**. **RPM** juga mendukung beberapa sistem operasi yang lain seperti **IBM AIX**, **Novell Netware**, **SGI IRIX**, **IRIX64** dan lain sebagainya. **RPM** menggunakan format paket **.rpm**;
3) **pkgtool**, digunakan oleh **Slackware Linux** (beserta turunannya) menggunakan format paket **.tgz** dan **.txz**;
4) **Pacman**, digunakan oleh **Arch Linux** (beserta turunannya), **KaOS** dan **Frugalware** (**pacman-g2**). Format paket yang digunakan adalah **.pkg.tar.xz** di **Arch Linux**, sedangkan **Frugalware** menggunakan format paket **.fpm**;
5) **Portage**, digunakan oleh **Gentoo Linux** yang serupa dengan **ports** (sistem manajemen paket yang digunakan **FreeBSD**);

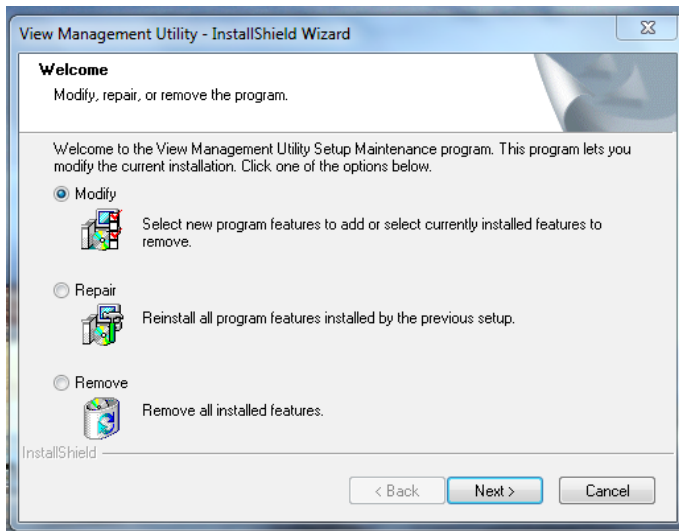
6) **Pisi**, asalnya digunakan oleh **Pardus Linux**, sekarang masih digunakan oleh distribusi Pisi Linux dan menggunakan format paket **.pisi**;
7) **conary**, digunakan oleh **Foresight Linux** yang sekarang sudah tidak dikembangkan lagi;
8) **Entropy**, digunakan oleh **Sabayon Linux** dan menggunakan format paket **.tbz2**;
9) **PETget**, digunakan oleh **Puppy Linux** dan menggunakan format paket **.pet**;
10) **eopkg**, digunakan oleh **Solus OS** (dulu bernama **Evolve OS**) dan menggunakan format paket **.eopkg**;
11) **pkgutils**, digunakan oleh **CRUX Linux** menggunakan format paket **.pkg.tar.gz**;
12) **snappy**, merupakan manajer paket masa depan **Ubuntu Linux** yang sekarang masih dalam tahap pengembangann dan menggunakan format paket **.snap**;
13) **Nix *Package Manager***, digunakan oleh **NixOS** dan menggunakan format paket **.nix**. Nix ini merupakan manajer paket fungsional murni (*purely functional*) yang memperlakukan paket seperti nilai-nilai dalam bahasa pemrograman fungsional murni seperti **Haskell**, memiliki dukungan terhadap banyak pengguna (*multi user*), peningkatan secara ***atomic*** dan pengembalian ke keadaan sebelumnya (*rollback*). **Nix** juga mengizinkan berbagai versi atau varian perangkat lunak dipasang pada waktu yang bersamaan. Selain **GNU/Linux**, **Nix** juga mendukung sistem operasi **Apple macOS**;
14) **Guix**, digunakan oleh **GuixSD**. **Guix** merupakan turunan dari **Nix *Package Manager***, yang menggunakan **Guile Scheme API** dan secara khusus menyediakan perangkat lunak bebas secara eksklusif;
15) dan lain sebagainya.

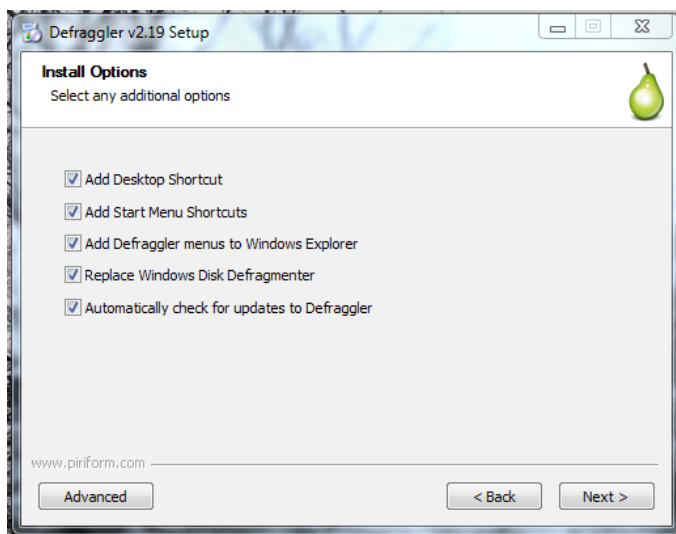
1.7. Sistem Manajemen Paket di Sistem Operasi Microsoft Windows

Jika kita berbicara tentang sistem manajemen paket, tentu tidaklah lengkap rasanya jika kita tidak membahas juga perihal sistem manajemen paket yang digunakan di sistem operasi komputer pribadi paling populer saat ini, yakni **Microsoft Windows**. Mungkin banyak di antara kita yang menganggap sistem operasi **Microsoft Windows** sampai saat ini tidak memiliki sistem manajemen paket atau perkakas manajer paket secara khusus seperti halnya pendahulunya yaitu sistem operasi **MS-DOS**. Karena seperti yang kita ketahui, menu **Programs and Features** atau **Add and Remove Program** yang terdapat di **Control Panel** memiliki fitur yang terbatas yaitu untuk menampilkan daftar aplikasi yang terpasang, menghapus aplikasi serta untuk mengaktifkan dan/atau menonaktifkan beberapa fitur dalam sistem operasi Windows. Selain itu untuk dapat melakukan pemasangan aplikasi, yang lebih kita perlukan adalah berkas *installer* dari aplikasi tersebut. Dan kemudian untuk memasangnya, kita hanya perlu melakukan klik dua kali terhadap berkas tersebut dan program *installer* yang merupakan bawaan di dalam berkas akan melakukan proses pemasangan aplikasi tersebut. Selama proses pemasangan program *installer* biasanya akan menampilkan wisaya atau *wizard* instalasi yang akan memandu pengguna dalam membantu pelaksanaan proses pemasangan, terutama untuk memberikan opsi-opsi tertentu yang dibutuhkan, seperti untuk memasukkan kode kunci lisensi (*lisences key*), menentukan kelengkapan aplikasi yang ingin dipasang, me-

nentukan lokasi target pemasangan, menjadikan sebagai aplikasi baku (*default*) sistem dan lain sebagainya.

Gambar 1.4: Add Remove Programs di Windows XP

Sebenarnya, jika kita mau untuk meneliti lebih mendalam tentang *installer* aplikasi yang digunakan di sistem operasi **Microsoft Windows**, ternyata terdapat dua golongan utama, yaitu:

1) *installer* yang menggunakan standar **Windows Installer**. Berkas paket installer dengan standar ini biasanya menggunakan ekstensi nama berkas .msi. Paket installer dapat dibangun menggunakan perkakas berikut:
    a) **Visual Studio Setup project**;
    b) **Windows Installer XML** (**WiX**), yang dibuat oleh **Outercurve Foundation** yang merupakan sebuah badan usaha nirlaba yang didirikan oleh **Microsoft**;
    c) **Install Shield**, yang dibuat oleh **Flexera Software**;
    d) **EMCO MSI Package Builder**, yang dibuat oleh **EMCO Software**;
    e) **Advanced Installer**, yang dibuat oleh **Caphyon Ltd**;
    f) dan lain sebagainya;

Gambar 1.5: Contoh Installer InstallShield

2) *installer* yang tidak mengikuti standar **Windows Installer**. Berkas paket installer golongan ini biasanya berupa berkas eksekutabel (**.exe**) yang format isi berkasnya mengikuti standar yang dibuat oleh pihak pengembang perkakas pembuat installernya maing-masing. Contoh perkakas pembuat installernya adalah:
   a) **Nullsoft Scriptable Install System** (**NSIS**) yang dibuat oleh **Nulsoft, Inc**. (perusahann pembuat pemutar multimedia **Winamp**);
   b) **Inno Setup**, yang dibuat oleh **Jordan Russell** dan **Martijn Laan**;
   c) dan lain sebagainya.

Gambar 1.6: Contoh Installer NSIS

Windows Installer atau yang dulu dikenal dengan Microsoft *Installer* adalah sebuah komponen dan **API** (*Application Programming Interface*) dari sistem

operasi **Microsoft Windows** yang digunakan untuk memasang, memelihara dan menghapus perangkat lunak. **Windows Installer** ini dapat digunakan oleh penerbit atau pengembang perangkat lunak untuk membantu membuat proses pemasangan dan penghapusan menjadi lebih konsisten dari satu program ke program lainnya. **Windows Installer** pertama kali dirilis pada tanggal 31 Agustus 1999 (versi 1.0), yang digunakan pertama kali pada *installer* **Microsoft Office 2000**. Saat ini, rilis terakhir adalah versi 5.0 yang dirilis bersama dengan rilis sistem operasi **Microsoft Windows 7** dan **Windows Server 2008 R2**. **Windows Installer** ini dapat kita anggap sebagai sistem manajemen paket yang digunakan di sistem operasi **Microsoft Windows**, walaupun tidak seperti sistem manajemen paket yang biasa kita temukan di distribusi **GNU/Linux**, **Windows Installer** tidak dapat mengatasi permasalahan dependensi. Secara umum, **Windows Installer** ini terdiri atas tiga komponen, yaitu:

1) Paket **Windows Installer**

   Paket **Windows Installer** sebenarnya menggunakan beberapa macam format ekstensi nama berkas, namun yang utama digunakan dalam proses pemasangan dan penghapusan perangkat lunak adalah paket yang menggunakan ekstensi nama berkas **.msi** (**Windows Installer Database**). **Windows Installer Database** atau berkas dengan ekstensi **.msi** adalah sebuah basis data relasional (*relational database*) yang menyimpan semua informasi, instruksi (perintah) dan data yang dibutuhkan untuk memasang dan menghapus suatu aplikasi dalam berbagai variasi skenario yang digunakan. Informasi yang terdapat di dalam berkas **.msi** ini terbagai ke dalam fitur dan komponen dan setiap komponen dapat membawa berkas-berkas, data *registry*, pintasan (shortcut) dan lain sebagainya. Berkas **.msi** juga berisi antarmuka untuk pengguna yang digunakan untuk instalasi (biasanya akan ditampilakan dalam bentuk wisaya atau *wizard*) dan berbagai data lain seperti prasyarat yang dibutuhkan, urutan prosedur instalasi dan lain sebagainya. Selain itu, berkas **.msi** ini juga dapat berisi berkas-berkas aktual yang akan dipasang, yang dapat berupa berkas kabinet (.cab), berkas yang tidak dikompres dan atau berkas transforms. Berkas-berkas aktual tersebut juga dapat diletakkan di luar dari paket selama dapat ditemukan oleh *installer*. Setelah dipasang, berkas **.msi** ini akan disimpan sebagai paket lokal dan telah diubah nama berkasnya dengan nama tertentu (biasanya) dalam direktori **C:\Windows\Installer** yang nantinya akan digunakan kembali saat pengguna akan melakukan penghapusan atau memperbaiki aplikasi tersebut. Berkas paket lokal ini nantinya juga akan didaftarkan dalam basis data **Windows Installer**. Ekstensi nama berkas yang digunakan oleh **Windows Installer** meliputi:

Tabel 1.2: Daftar Ekstensi Nama Berkas Windows Installer

Ekstensi	**Deskripsi**
.msi	Windows Installer Database
.msm	Windows Installer Merge Module
.msp	Windows Installer Patch
.mst	Windows Installer Transform
.idt	Exported Windows Installer Database Table

Ekstensi	**Deskripsi**
.cub	Validation module
.pcp	Windows Installer Patch Creation File

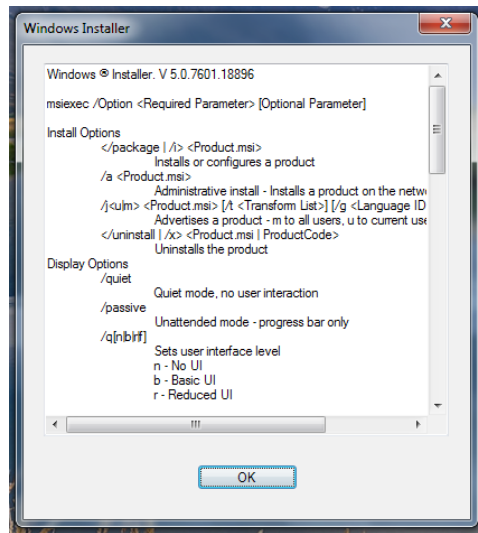
2) **msiexec.exe**
Berkas **.msi** bukan merupakan berkas eksekutabel seperti *installer* lain yang menggunakan ekstensi .exe, sehingga untuk dapat menjalankannya dibutuhkan perkakas khusus. Perkakas tersebut adalah **msiexec.exe**, yang dapat kita temukan di direktori **C:\Windows\System32**. Perkakas **msiexec.exe** ini menyediakan sarana untuk menginstal, memodifikasi dan melakukan operasi pada **Windows Installer** dari baris perintah (*command line*). Perkakas inilah yang berperan sebagai perkakas manajer paket. Walaupun merupakan perkakas baris perintah, setiap kita menjalankan berkas **.msi** dalam mode grafis (seperti saat kita melakukan klik dua kali terhadap berkas **.msi** melalui **Windows Explorer**), sistem operasi tetap akan memanggil **msiexec.exe** ini dan menjalankannya di latar belakang. Hal ini dapat kita periksa dengan menggunakan perkakas **Task Manager**. Untuk menjalankan perkakas ini kita dapat menggunakan berbagai opsi perintah yang tersedia, namun kita tidak akan membahas semua opsi yang ada tersebut. Opsi baris perintah **Windows Installer** ini tidak case-sensitive. Beberapa opsi dari perintah **msiexec.exe** antara lain:

Tabel 1.3: Daftar Opsi Perintah msiexec.exe

Opsi	*Parameter*	*Keterangan dan Contoh*

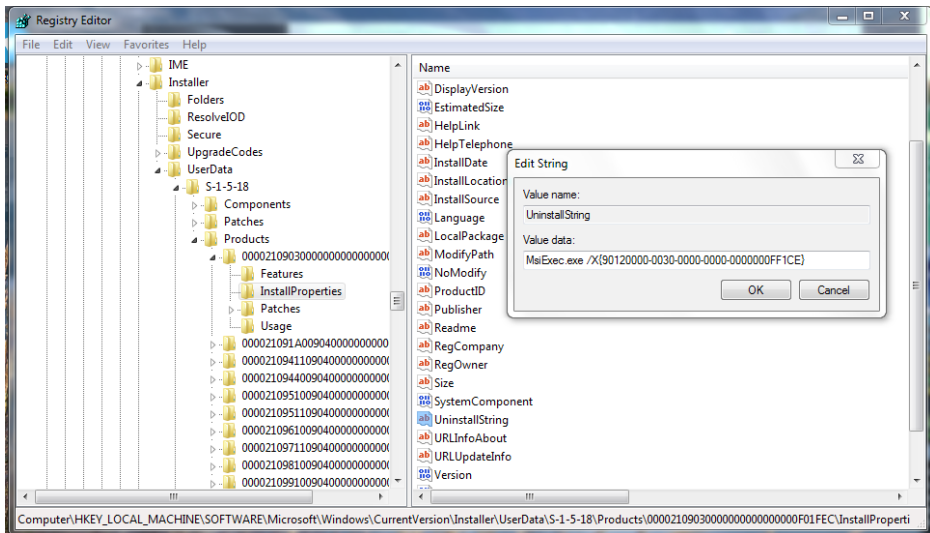
`/i`	`Paket|Kode Produk`	Memasang atau mengkonfigurasi produk
		Contoh:
		`msiexec /i LibreOffice_5.1.0_Win_x86.msi`
		`msiexec /i {2F75F86D-8362-4F49-9536-D87DCB-F6ABAE}`

`/f`	`[p|o|e|d|c|a|u|m|s|v] Paket|Kode Produk`	Memperbaiki produk. Daftar argumen baku untuk opsi ini adalah “omus”. p – memasang ulang hanya jika berkas hilang o – memasang ulang jika berkas hilang atau versi yang lebih lama terpasang e – memasang ulang jika berkas hilang atau versi yang sama atau lebih lama terpasang d – memasang ulang jika berkas hilang atau versi yang berbeda terpasang e – memasang ulang jika berkas hilang atau checksum yang disimpan tidak sesuai dengan nilai yang dihitung a – memaksa semua berkas untuk dipasang ulang u – menulis ulang semu entri *registry* yang dibutuhkan pengguna m – menulis ulang semua entri *registry* yang dibutuhkan komputer s – menimpa semua pintasan (shortcut) yang ada v – menjalankan dari sumber atau melakukan cache ulang paket lokal Contoh:

Opsi	Parameter	Keterangan dan Contoh
		msiexec /fomus {2F75F86D-8362-4F49-9536-D87DCB-F6ABAE}
/x	Paket|Kode Produk	Menghapus produk msiexec /x LibreOffice_5.1.0_Win_x86.msi msiexec /x C:\Windows*Installer*\15ed133.msi msiexec /x {2F75F86D-8362-4F49-9536-D87DCB-F6ABAE}
/p	Paket Patch[;paket patch2...]	Menerapkan Patch
/q	n|b|r|f|n+|b+|b-	Mengatur level antarmuka /q, /qn – tanpa antarmuka /qb – antarmuka dasar. Kita dapat menggunakan /qb! untuk menyembunyikan tombol Batal (Cancel) /qr – antarmuka yang dikurangi tanpa modal dialog box di akhir proses instalasi /qf – antarmuka penuh, mengizinkan modal dialog box fatal Error, UserExit atau Exit di akhir /qn+ - tanpa antarmuka kecuali untuk modal dialog box yang ditampilkan di akhir /qb+ - antarmuka dasar dengan modal dialog box di-tampilkan di akhir /qb- - antarmuka dasar tanpa modal dialog box Contoh: msiexec /q /i LibreOffice_5.1.0_Win_x86.msi
/a	Paket	Opsi pemasangan administratif. Memasang produk dalam jaringan Contoh: msiexec /a LibreOffice_5.1.0_Win_x86.msi
/?|/h		Menampilkan informasi dari Windows Installer

Gambar 1.7: Tampilan bantuan program msi-exec.exe

3) Basis data

   Basis data yang dimaksud di sini adalah tempat disimpannya informasi yang berhubungan dengan paket perangkat lunak yang terpasang di sistem operasi. **Windows Installer** menyimpan informasi-informasi tentang paket tersebut di beberapa tempat di *registry* Windows. Beberapa lokasi di *registry* yang kami ketahui sebagai tempat penyimpanan informasi tentang paket perangkat lunak yang terpasang antara lain:

   a) **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion*Uninstall***, berisi informasi tentang semua aplikasi yang terinstall, baik yang *installer*nya menggunakan standar **Windows Installer** maupun tidak. Informasi yang disimpan biasanya berupa nama aplikasi, versi, icon, nama pengembang, letak aplikasi terpasang, ***UninstallString***, tanggal pemasangan dan lain sebagainya;
   b) **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion***Installer***\Folders**, berisi daftar direktori tempat aplikasi yang terpasang khusus *installer*nya yang menggunakan standar **Windows Installer**;
   c) **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion***Installer***\UserData**, berisi semua informasi dari aplikasi yang menggunakan *installer* dengan standar **Windows Installer**. Informasi yang disimpan berupa daftar semua komponen aplikasi, *patch* dan informasi tentang produk (aplikasi yang bersangkutan), seperti letak paket lokal tersimpan dan informasi lain seperti yang juga terdapat dalam **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion*Uninstall***.

Gambar 1.8: Contoh entri registry milik aplikasi di Windows 7

Di sistem operasi **Microsoft Windows**, aplikasi biasanya akan dipasang di direktori tertentu sesuai dengan standar yang telah ditetapkan. Untuk aplikasi sistem biasanya akan dipasang di direktori **C:\Windows\System32**. Sedangkan aplikasi tambahan atau aplikasi pihak ketiga akan dipasang di direktori **C:\Program Files** atau **C:\Program Files (x86)** untuk aplikasi yang dibuat untuk arsitektur 32 bit yang terpasang di sistem operasi dengan arsitektur 64 bit (x86_64). Walaupun begitu, sistem sebenernya juga tidak melarang jika ada aplikasi yang dipasang di tempat lain seperti di **C:\Folder** maupun di tempat lain sesuai kebutuhan pengembang aplikasi.

BAB 2

SISTEM MANAJEMEN PAKET DEBIAN

2.1. Sejarah Proyek Debian

Sebelum kita membahas tentang **Sistem Manajemen Paket Debian**, tidak lah lengkap rasanya jika kita tidak menyinggung mengenai sejarah **Proyek Debian** ini terlebih dahulu. Hal ini dikarenakan adanya kaitan yang erat antara **Sistem Manajemen Paket Debian** ini dengan **Proyek Debian** itu sendiri. **Proyek Debian** adalah sebuah kelompok sukarelawan dari seluruh dunia yang berusaha untuk menghasilkan distribusi sistem operasi yang seluruhnya tersusun dari perangkat lunak bebas. Produk utama dari proyek sampai saat ini adalah **Distribusi Debian GNU/Linux**, yang meliputi **kernel sistem operasi Linux** dan ribuan aplikasi *pre-packaged*.

Proyek Debian secara resmi didirikan oleh **Ian Murdock** pada 16 Agustus 1993, saat beliau masih menjadi mahasiswa di **Universitas Purdue**, Amerika Serikat. Nama **Debian** berasal dari gabungan nama sang pembuat yaitu **Ian Murdock** dan kekasihnya saat itu yaitu *Debra Lynn*. Saat itu, keseluruhan konsep dari sebuah distribusi **Linux** merupakan sesuatu hal yang baru. **Ian** berangan-angan menjadikan **Debian** sebagai distribusi yang dibuat secara terbuka, sesuai dengan semangat **Linux** dan **GNU**.

Pada saat proyek dimulai, **Debian** merupakan satu-satunya distribusi yang membuka diri bagi setiap pengembang dan pengguna untuk berkontribusi pada pekerjaannya. **Debian** merupakan sedikit dari distribusi **GNU/Linux** yang paling penting yang bukan merupakan sebuah entitas komersial. **Debian** merupakan satu-satunya proyek besar yang memiliki dokumen tentang konstitusi, kontrak sosial (disebut sebagai **Debian Social Contract**) dan aturan-aturan yang mengatur proyek tersebut. **Debian** ini juga merupakan satu-satunya distribusi yang paketnya memiliki informasi ketergantungan (dependensi) secara rinci yang memperhatikan hubungan antar paket sehingga dapat menjamin konsistensi sistem setelah dilakukan peningkatan (*upgrade*).

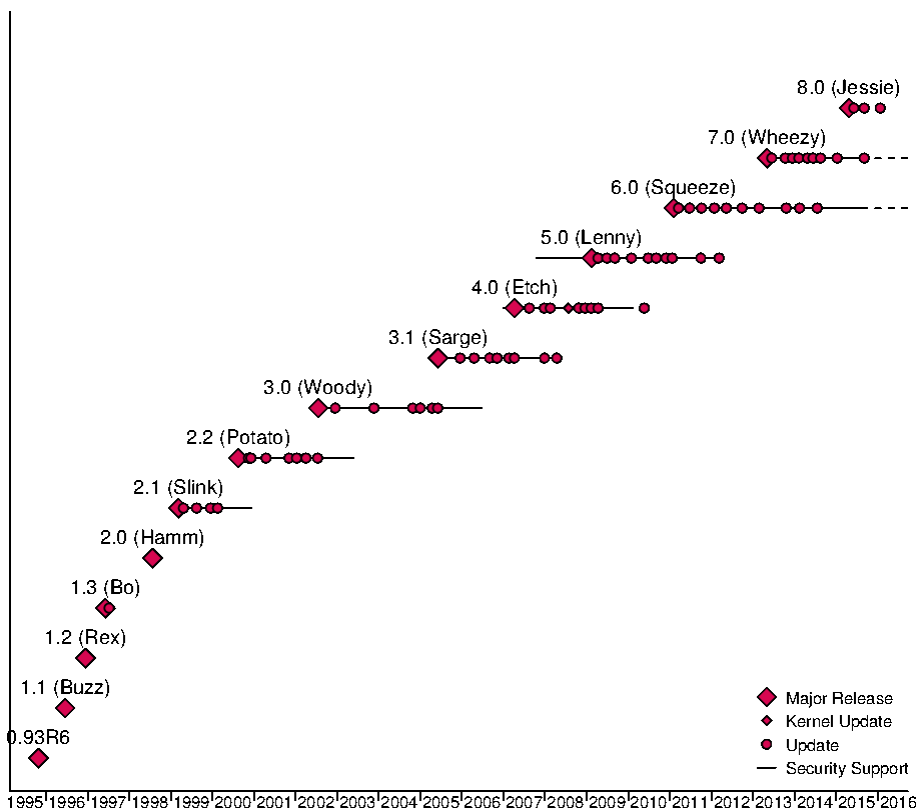
Untuk mencapai dan menjaga standar kualitas yang tinggi, **Debian** telah mengadopsi sekumpulan aturan dan prosedur dalam memaketkan dan menyampaikan perangkat lunak. Standar ini didukung dengan alat-alat, otomasi dan dokumen yang mengimplementasikan semua elemen kunci **Debian** dalam cara yang terbuka dan nyata. Berikut sejarah singkat dari rilis **Debian** dari awal berdirinya sampai dengan saat buku ini disusun, yaitu:

1) **Debian 0.01** (pre-ALPHA) dirilis pada tanggal 15 September 1993, yang merupakan yang pertama dari sejumlah rilis internal yang dirilis pada periode bulan Agustus sampai dengan Januari 1994. Beberapa rilis internal tersebut adalah **Debian 0.02** ALPHA (17 Oktober 1993), **Debian 0.03** ALPHA (2 No-

vember 1993), **Debian 0.04** ALPHA (7 November 1993), **Debian 0.80** BETA (23 November 1993), **Debian 0.81** BETA (28 November 1993) dan **Debian 0.90** BETA (26 Januari 1994).

2) **Debian 0.91** BETA dirilis pada tanggal 29 Januari 1994. Rilis ini telah memiliki sistem paket sederhana yang dapat digunakan untuk memasang dan menghapus paket (cikal bakal dari **dpkg**).
3) **Debian 0.93R5** (Maret 1995). Pada rilis ini, tanggung jawab atas masing-masing paket telah secara jelas ditugaskan pada masing-masing pengembang dan manajer paket **dpkg** telah digunakan untuk memasang paket setelah dilakukan pemasangan sistem dasar.
4) **Debian 0.93R6** (26 Oktober 1995) memperkenalkan perkakas **dselect**. Rilis ini merupakan rilis **Debian** terakhir yang menggunakan format biner **a.out** dan dipelihara oleh 60 pengembang. Pertama kali peladen (*server)* **master.debian.org** dibuat oleh **Bdale Garbee**.
5) **Debian 1.0** tidak pernah dirilis. Hal ini dikarenakan ada masalah penamaan dengan pihak vendor CD.
6) **Debian 1.1** ***Buzz*** (17 Juni 1996). Rilis ini merupakan rilis pertama **Debian** yang menggunakan nama kode (*code name*). Pada saat itu, **Bruce Perens** mengambil alih kepemimpinan proyek dari **Ian Murdock**. Rilis ini telah mendukung format biner **ELF** secara penuh, menggunakan kernel **Linux** 2.0 dan berisi 474 paket.
7) **Debian 1.2** ***Rex*** (12 Desember 1996). Rilis ini berisi 848 paket yang dipelihara oleh 120 pengembang.
8) **Debian 1.3** ***Bo*** (5 Juni 1997). Rilis ini berisi 974 paket yang dipelihara oleh 200 pengembang.
9) **Debian 2.0** ***Hamm*** (24 Juli 1998). Rilis ini merupakan rilis pertama **Debian** yang mendukung banyak arsitektur (*multi-architecture*), menambahkan dukungan untuk arsitektur **Motorola** seri **6800** (**m68k**) dan membuat transisi ke **libc**.
10) **Debian 2.1** ***Slink*** (9 Maret 1999). Rilis ini menambahkan dukungan untuk arsitektur **Alpha** dan **SPARC**. Inovasi kunci pada rilis ini adalah diperkenalkannya **APT** sebagai antarmuka manajemen paket yang baru. **APT** ini telah membuat sebuah paradigma baru dalam mendapatkan dan pemasangan paket pada sistem operasi **open source**.
11) **Debian 2.2** ***Potato*** (15 Agustus 2000). Rilis ini menambahkan dukungan untuk arsitektur **PowerPC** dan **ARM**.
12) **Debian 3.0** ***Woody*** (19 Juli 2002). Rilis ini menambahkan dukungan terhadap lebih banyak arsitektur, yaitu **IA-64**, **HP PA-RISC**, **MIPS** (big endian), **MIPS** (little endian) dan **S/390**. Rilis ini juga rilis pertama yang menyertakan perangkat lunak kriptografi dan desktop **KDE** (setelah masalah lisensi dengn **Qt** dapat diselesaikan).
13) **Debian 3.1** ***Sarge*** (6 Juni 2005). Tidak ada penambahan arsitektur baru pada rilis ini, meskipun porting tidak resmi **AMD64** telah diperkenalkan pada waktu yang sama dan didistribusikan melalui situs hosting proyek **Alioth** yang baru. Rilis ini menampilkan program installer baru yaitu **debian-installer**, sebuah perangkat lunak modular yang memiliki pendeteksian perangkat keras secara otomatis dan dirilis dengan diterjemahkan secara penuh ke lebih dari tiga puluh bahasa. Rilis ini merupakan rilis pertama yang menyertakan perangkat lunak perkantoran lengkap **OpenOffice.org**.
14) **Debian 4.0** ***Etch*** (8 April 2007). Rilis ini menambahkan dukungan terhadap arsitektur **AMD64** (**x86_64**) dan menghentikan dukungan resmi untuk arsi-

tektur **m68k**. Rilis ini meneruskan penggunaan **debian-installer**, namun dengan tambahan antarmuka grafis, verifikasi dengan metode kriptografi atas paket yang diunduh,pembuatan partisi yang lebih fleksibel (dengan dukungan untuk partisi yang dienkripsi), konfigurasi email yang disederhanakan, pemilihan *desktop* yang lebih fleksibel, lokalisasi yang disederhanakan namun lebih baik dan mode baru, termasuk sebuah mode penyelamatan (*rescue mode*).

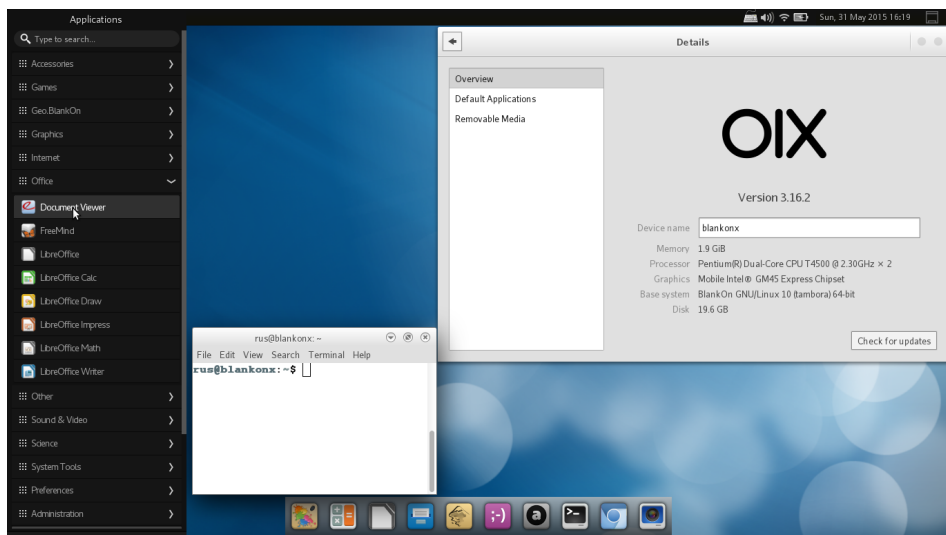
15) **Debian 5.0 *Lenny*** (14 Pebruari 2009). Rilis ini menambahkan dukungan terhadap arsitektur **ARM EABI** (**armel**) yang menyediakan dukungan untuk prosesor **ARM** yang lebih baru dan menghentikan dukungan untuk arsitektur **ARM** yang lama (**arm**). Rilis ini juga merupakan rilis pertama yang menyediakan versi bebas dari teknologi **Sun Java** (**OpenJDK**).
16) **Debian 6.0 *Squeeze*** (6 Pebruari 2011). Rilis ini menghilangkan dukungan untuk arsitektur **hppa** dan **alpha**. Dua arsitektur baru yang merupakan porting dari **FreeBSD** (**kfreebsd-i386** dan **kfreebsd-amd64**)telah dibuat sebagai *technology preview*.
17) **Debian 7.0 *Wheezy*** (4 Mei 2013). Rilis ini menambahkan dukungan terhadap arsitektur **armhf** dan **s390x** (**IBM ESA/**390 atau userland 64 bit untuk komputer mainframe **IBM System Z**) dan memperkenalkan dukungan terhadap **multi-arch** yang memperkenankan pengguna untuk memasang paket dari berbagai arsitektur dalam mesin yang sama. Rilis ini juga rilis pertama yang mendukung pemasangan dan *booting* dalam perangakat yang menggunakan *firmware* **UEFI**.
18) **Debian 8.0 *Jessie*** (25 April 2015). Rilis ini menembakan dukungan terhadap arsitektur **arm64** (**AArch64**) dan **PowerPC 64 bit litte endian** (**ppc64el**), namun menghentikan dukungan untuk *porting* arsitektur **ia64**, **s390**, **kfreebsd-i386** dan **kfreebsd-amd64**. Rilis ini mengganti **init system** baku dari **sysvinit** ke **systemd**.
19) **Debian 9.0 *Stretch*** yang merupakan rilis selanjutnya dari sistem operasi **Debian** (tanggal rilis belum ditentukan).

Gambar 2.1: Grafik Garis Waktu Rilis Debian
Sumber: http://wiki.debian.org/DebianReleases

2.2. Mengenal Sistem Manajemen Paket Debian

Seperti yang telah kita ketahui sebelumnya, di lingkungan sistem operasi **GNU/Linux** terdapat berbagai macam sistem manajemen paket. Dari sekian banyak sistem manajemen paket yang ada tersebut, terdapat beberapa sistem manajemen paket yang secara luas telah digunakan dan diadopsi di banyak distribusi **GNU/Linux** yang ada saat ini. Salah satu di antara sistem manajemen paket tersebut adalah **Sistem Manajemen Paket Debian**.

Sistem Manajemen Paket Debian pada awalnya merupakan sistem manajemen paket yang dibuat untuk distribusi **Debian GNU/Linux**. Namun saat ini, **Sistem Manajemen Paket Debian** telah digunakan secara luas di banyak distribusi **GNU/Linux**, bahkan juga bisa dianggap sebagai sistem manajemen paket yang paling populer dan memiliki basis pengguna paling besar jika dibandingkan dengan sistem manajemen paket lainnya. Beberapa distribusi tersebut di antaranya adalah:

1) **Debian**, utamanya adalah rilis utama yang menggunakan *kernel* **Linux** (**Debian GNU/Linux**), termasuk beberapa *port* **Proyek Debian** yang menggunakan *kernel* sistem operasi lain, seperti **Debian GNU/kFreeBSD** dan **Debian GNU/Hurd**;

2) **Ubuntu Linux**, termasuk berbagai turunannya, seperti **Kubuntu**, **Xubuntu**, **Lubuntu**, **Edubuntu**, **Ubuntu GNOME**, **Ubuntu MATE**, **Ubuntu Kylin**, **Ubuntu Studio** dan **Mythbuntu**;
3) **Linux Mint** dan semua variannya, termasuk rilis **LMDE** (**Linux Mint Debian Edition**);
4) **Knoppix**;
5) **MEPIS**, termasuk **antiX**;
6) **Kali Linux**;
7) **Trisquel GNU/Linux**;
8) **Gnewsense GNU/Linux**;
9) **Blankon Linux**;
10) **Devuan**;
11) **Pardus**;
12) **Parsix GNU/Linux**;
13) **Siduction**;
14) berbagai turunan **Ubuntu Linux**, seperti **Linux Lite**, **LXLE**, **Elementary OS**, **Zorin OS** dan lain sebagainya;
15) **Linux Deepin**;
16) dan lain sebagainya.

Gambar 2.2: Blankon GNU/Linux

Seperti halnya sistem manajemen paket yang lain, **Sistem Manajemen Paket Debian** ini terdiri atas beberapa komponen. Komponen-komponen tersebut adalah:

1) Paket perangkat lunak, yang dikenal dengan sebutan “paket **debian**“ (**debian *package***). Paket **debian** terbagi atas dua tipe, yaitu:
   a) Paket biner;
   b) Paket sumber;
2) Perkakas manajer paket, yang terdiri atas beberapa tingkatan, yaitu:
   a) **dpkg**, sebagai perkakas manajemen paket pada tingkatan atau lapisan paling dasar dan merupakan inti dari **Sistem Manajemen Paket Debian** ini. Perkakas ini adalah perkakas utama yang dapat bekerja secara langsung terhadap paket;

b) **APT**, sebagai perkakas manajemen paket tingkat lanjut yang membantu mengatasi permasalahan dependensi dan mendapatkan paket yang diminta;

c) berbagai perkakas manajemen paket yang lain, yang merupakan *front-end* dari **dpkg** dan **APT**, seperti **dselect**, **taskel**, **aptitude**, **wajig**, **gdebi**, **synaptic** dan lain sebagainya;

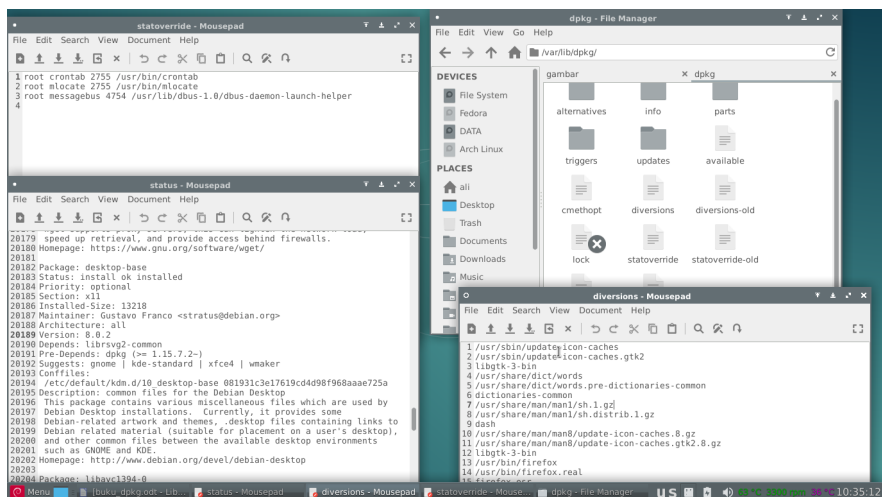
3) Basis data, yang berfungsi sebagai tempat penyimpanan informasi yang berhubungan dengan paket. Basis data tersebut dapat terdiri atas beberapa berkas dan disimpan di dalam tempat tertentu yang telah ditentukan. Di antara tempat penyimpanan berkas basis data dan beberapa contoh berkas basis data tersebut adalah:

a) **/var/lib/dpkg/**, merupakan direktori tempat penyimpanan berkas basis data yang digunakan oleh manajer paket **dpkg**. Di dalam direktori tersebut terdapat beberapa berkas, misalnya:

i) **/var/lib/dpkg/available**, berisi daftar paket yang tersedia dan dapat dipasang ke dalam sistem;

ii) **/var/lib/dpkg/status**, merupakan status dari paket-paket yang tersedia yang berisi informasi tentang apakah paket ditandai untuk dihapus atau tidak, apakah paket telah terpasang atau tidak dan lain sebagainya. Sedangkan berkas **/var/lib/dpkg/status-old**, merupakan berkas pencadangan dari berkas **/var/lib/dpkg/status**. Selain itu, berkas ini juga dicadangkan setiap hari dalam direktori **/var/backups/** yang berguna ketika berkas tersebut hilang atau rusak dikarenakan masalah pada sistem berkas (*filesystem*);

iii) **/var/lib/dpkg/statoverride**, yang berisi daftar terkini dari perencanaan penolakan sistem;

iv) **/var/lib/dpkg/diversions**, yang berisi daftar terkini dari pengalihan sistem;

Gambar 2.3: Contoh berkas basis data di Sistem Manajemen Paket Debian

b) **/var/lib/apt/**, merupakan direktori tempat penyimpanan berkas basis data yang digunakan oleh manajer paket **APT**. Di dalam direktori ini terdapat beberapa direktori yang digunakan untuk menyimpan bagian informasi yang berbeda, misalnya:

i) **/var/lib/apt/lists/**, merupakan tempat penyimpanan informasi dari setiap paket yang secara spesifik berasal dari repositori yang terdaftar dalam berkas **sources.list**; dan
ii) **/var/lib/apt/partial/**, merupakan tempat penyimpanan informasi dalam perlintasan (sementara).

3.1. Sekilas tentang Paket Debian

Paket **Debian** merupakan bentuk standar untuk berkas paket perangkat lunak yang awalnya dirancang untuk digunakan oleh sistem operasi yang berasal dari **Proyek Debian** seperti **Debian GNU/Linux**, **Debian GNU/Hurd** dan **Debian GNU/kFreeBSD**, dan kemudian juga digunakan oleh sistem operasi lain yang memilih untuk turut serta menggunakan **Sistem Manajemen Paket Debian** (terutama oleh sistem operasi atau distribusi **GNU/Linux** yang merupakan turunan atau yang pengembangannya berdasar pada distribusi **Debian GNU/Linux**). Paket **Debian** terdiri atas dua tipe, yaitu paket biner (*binary packages*) dan paket kode sumber (*source packages*).

3.2. Paket Biner Debian

Paket biner **Debian** (bisa juga disebut dengan berkas arsip **Debian**) atau yang lebih umum dikenal dengan sebutan “paket **Debian**” saja ini merupakan bentuk atau format standar dari berkas paket perangkat lunak *pre-compiled* yang digunakan dalam **Sistem Manajemen Paket Debian**, yang di dalamnya dapat berupa berbagai berkas biner eksekutabel, *script*, pustaka (*library*), berkas konfigurasi, manual/halaman info, informasi hak cipta, dokumentasi dan berkas-berkas lainnya yang berhubungan dengan paket atau perangkat lunak yang bersangkutan. Versi terakhir dari format paket biner **Debian** ini adalah versi 2.0, yang diperkenalkan sejak **dpkg 0.93.76** dan dibuat sebagai *default* sejak **dpkg 1.2.0** dan rilis **Debian 1.1.1elf** (**i386/ELF** *builds*). Berkas paket biner **Debian** ini menggunakan ekstensi nama berkas **.deb**.

Nama berkas paket biner **Debian** harus memenuhi kaidah atau ketentuan sebagaimana berikut:

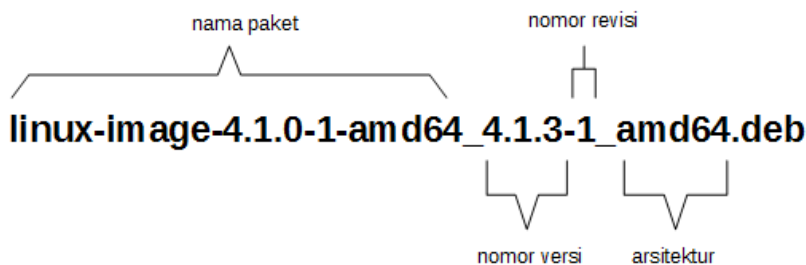
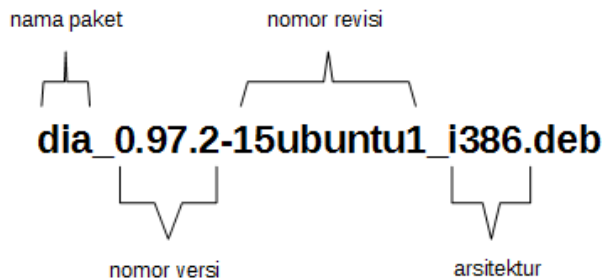
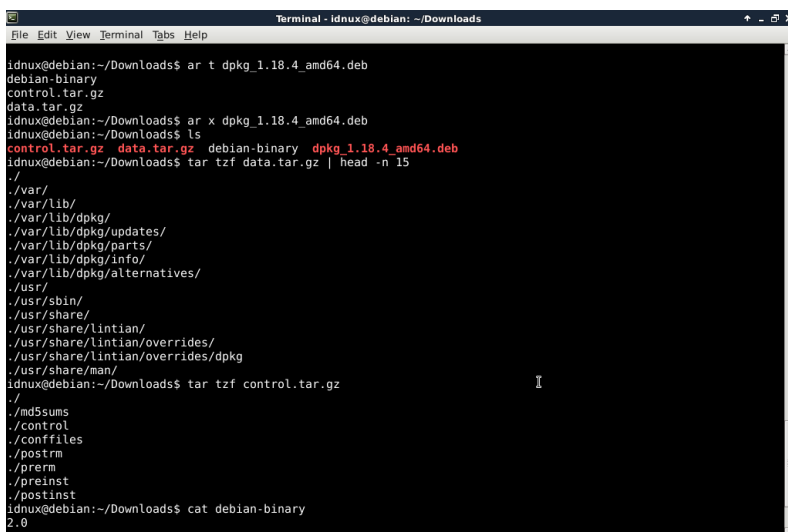
```
[epoch]:[package name]_[version number]-[Debian revision
number]_[architecture].deb
```

Penjelasan:

1) **epoch**, merupakan bilangan bulat (*integer*) tunggal yang disediakan untuk memperkenankan kekeliruan dalam nomor versi dari paket versi lama dan juga untuk menghilangkan atau mengganti skema penomoran versi sebelumnya yang digunakan paket. Komponen ini boleh diabaikan atau dihilangkan dari skema penomoran paket.
2) **package name**, merupakan nama dari paket atau perangkat lunak.
3) **version number** atau **upstream version number**, merupakan nomor versi yang secara spesifik ditentukan oleh pihak pengembang hulu (*upstream*)

dan tidak ada format standar atas nomor versi ini (nomor versi ini mungkin memiliki format yang berbeda seperti “20160130” atau “1.0.1pre1”).

4) **Debian revision number**, merupakan nomor yang ditentukan oleh pengembang **Debian** (atau oleh pengguna pribadi yang memilih untuk membangun sendiri paket). Nomor ini sesuai dengan tingkat revisi pada paket **Debian** tersebut, yang berarti tingkat revisi baru yang biasanya menandakan perubahan dalam **Debian Makefile** (**debian/rules**), berkas **control Debian** (**debian/control**), *script* pemasangan atau penghapusan (seperti **debian/postinstall**, **debian/pre-remove** dan lain sebagainya), atau dalam berkas konfigurasi yang digunakan bersama paket.
5) **architecture** atau arsitektur, menunjukkan nama tipe perangkat keras tertentu yang ditujukan sebagai target dari hasil kompilasi paket.

Contoh:

Gambar 3.1: Contoh format nama berkas Paket Debian

Gambar 3.2: Contoh nama berkas Paket Debian di Ubuntu Linux

Selain tersedia dalam format berkas dengan ekstensi nama berkas **.deb** yang biasa kita kenal, beberapa paket inti **Debian** juga tersedia dalam format paket **micro deb**. Paket dalam format ini biasanya digunakan untuk *bootstraping* pemasangan sistem **Debian**. Paket **micro deb** ini menggunakan ekstensi nama berkas **.udeb**.

3.2.1. Struktur Paket Biner Debian

Bentuk paket biner **Debian** dirancang agar isinya dapat diekstrak di setiap sistem operasi **Unix** maupun mirip **Unix** yang memiliki perkakas atau perintah klasik **ar**, **tar** dan **gzip** (terkadang juga menggunakan **xz** atau **bzip2**). Hal yang terlihat sepele ini penting untuk *portabilitas* dan saat pemulihan dari kerusakan.

Sebagai contoh, andaikan kita melakukan kesalahan dengan menghapus program **dpkg**, sehingga akan menyebabkan kita tidak dapat lagi memasang paket **Debian** ke dalam sistem. Perkakas **dpkg** ini sendiri juga tersedia dalam bentuk paket **Debian**, sehingga kita tentu tidak akan dapat memasangnya menggunakan cara yang biasa. Untungnya, ternyata masih ada cara lain yang dapat kita lakukan, yaitu dengan mengunduh berkas paket **dpkg** tersebut dan memasangnya secara manual (menggunakan cara penyalinan). Dan jika ternyata kesalahan yang telah kita lakukan tersebut menyebabkan satu atau lebih dari program **ar**, **tar**, atau **gzip/xz/bzip2** menghilang, kita hanya perlu menyalin program yang hilang dari sistem lain (karena masing-masing program beroperasi secara otonom dan tanpa ketergantungan).

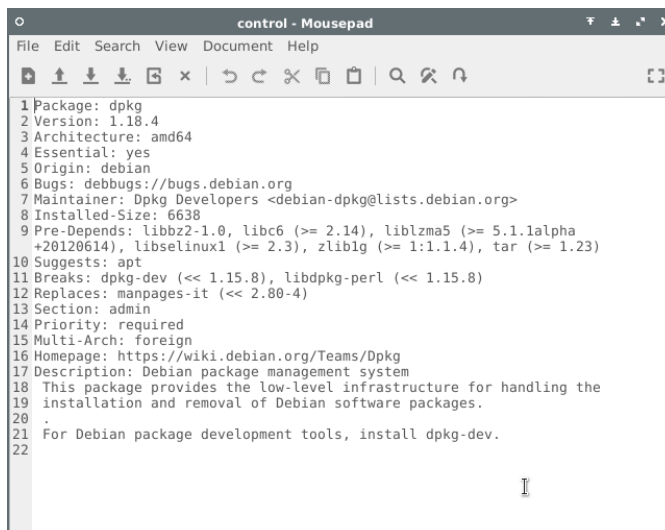
Sebagai contoh, kita dapat melihat isi dari berkas paket biner **Debian** dengan mengikuti beberapa langkah dalam gambar berikut:

```
Terminal - idnux@debian: ~/Downloads
File Edit View Terminal Tabs Help

idnux@debian:~/Downloads$ ar t dpkg_1.18.4_amd64.deb
debian-binary
control.tar.gz
data.tar.gz
idnux@debian:~/Downloads$ ar x dpkg_1.18.4_amd64.deb
idnux@debian:~/Downloads$ ls
control.tar.gz  data.tar.gz  debian-binary  dpkg_1.18.4_amd64.deb
idnux@debian:~/Downloads$ tar tzf data.tar.gz | head -n 15
./
./var/
./var/lib/
./var/lib/dpkg/
./var/lib/dpkg/updates/
./var/lib/dpkg/parts/
./var/lib/dpkg/info/
./var/lib/dpkg/alternatives/
./usr/
./usr/sbin/
./usr/share/
./usr/share/lintian/
./usr/share/lintian/overrides/
./usr/share/lintian/overrides/dpkg
./usr/share/man/
idnux@debian:~/Downloads$ tar tzf control.tar.gz
./
./md5sums
./control
./conffiles
./postrm
./prerm
./preinst
./postinst
idnux@debian:~/Downloads$ cat debian-binary
2.0
```

Gambar 3.3: Contoh isi Paket Debian

Seperti yang dapat kita lihat, di dalam berkas paket biner **Debian** tersebut terdapat tiga macam berkas, yaitu:

1) **debian-binary**. Berkas ini merupakan sebuah berkas yang hanya menunjukkan versi dari format berkas paket biner **Debian** yang digunakan (versi terakhir sampai saat buku ini disusun adalah versi 2.0).
2) **control.tar.gz**. Berkas arsip ini memiliki semua meta-data yang tersedia, seperti nama, versi paket dan lain sebagainya. Adanya meta-data ini akan membantu perkakas manajemen paket dalam melakukan kegiatan manajemen paket.
3) **data.tar.gz**. Berkas arsip ini berisi semua berkas yang akan diekstrak dari paket seperti semua berkas eksekutabel, dokumentasi dan berkas lainnya.

Beberapa paket dapat menggunakan format kompresi yang lain, sehingga berkas akan diberikan dengan nama yang berbeda (**data.tar.bz2** untuk **bzip2**, **data.tar.xz** untuk **xz** dan **data.tar.lzma** untuk **LZMA**).

3.2.2. Meta-data Paket

Berkas paket **Debian** tidak hanya berisi kumpulan berkas yang ditujukan untuk dipasang ke dalam sistem, namun juga berisi sekumpulan berkas yang menyediakan meta-data tambahan tentang paket yang juga dikenal dengan sebutan berkas "*control information files*" atau dikenal juga sebagai berkas-berkas meta data paket. Berkas "*control information files*" ini di antaranya adalah berkas "**control**", *script* pemelihara paket (*package maintainer scripts*), berkas "**conffiles**", berkas *checksums*, berkas "**list**" serta berkas "**symbols**" dan berkas "**shlibs**". Semua berkas-berkas ini disimpan dalam berkas arsip **control.tar.gz** yang terdapat di dalam paket.

Setelah paket dipasang, berkas-berkas meta-data atau "*control information files*" milik paket ini akan disimpan dalam direktori **/var/lib/dpkg/info/**. Berkas yang bersangkutan atau relevan pada suatu paket akan dinamai dengan nama paket yang bersangkutan dan memiliki ekstensi nama berkas mulai dari "**preinst**", "**postinst**" dan lain sebagainya menurut ketentuan yang sesuai. Contoh **foo.preinst**, **foo.postrm** dan lain sebagainya. Di dalam direktori ini juga dapat kita temukan berkas dengan ekstensi nama berkas **.list** untuk masing-masing paket yang berisi daftar semua berkas yang dipasang bersama paket yang bersangkutan.

3.2.2.1. Gambaran tentang Berkas "control"

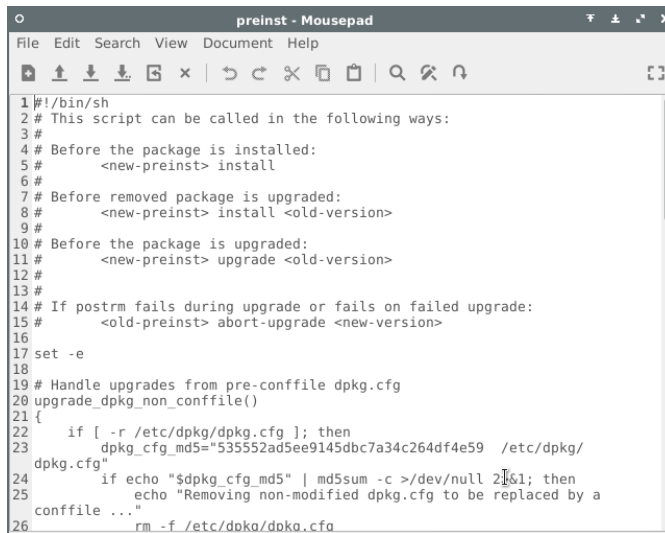
Berkas **control** berisi tentang berbagai nilai atau informasi yang digunakan oleh perkakas manajer paket (seperti **dpkg**, **APT**, **aptitude**, **synaptic** dan lain sebagainya) dalam melakukan pengelolaan paket. Sebagai contoh, berikut ini merupakan isi berkas **control** milik paket **dpkg**:

```
Package: dpkg
Version: 1.18.4
Architecture: amd64
Essential: yes
Origin: debian
Bugs: debbugs://bugs.debian.org
Maintainer: Dpkg Developers <debian-dpkg@lists.debian.org>
Installed-Size: 6638
Pre-Depends: libbz2-1.0, libc6 (>= 2.14), liblzma5 (>= 5.1.1alpha
+20120614), libselinux1 (>= 2.3), zlib1g (>= 1:1.1.4), tar (>= 1.23)
Suggests: apt
Breaks: dpkg-dev (<< 1.15.8), libdpkg-perl (<< 1.15.8)
Replaces: manpages-it (<< 2.80-4)
Section: admin
Priority: required
Multi-Arch: foreign
Homepage: https://wiki.debian.org/Teams/Dpkg
Description: Debian package management system
 This package provides the low-level infrastructure for handling the
 installation and removal of Debian software packages.
 .
 For Debian package development tools, install dpkg-dev.
```

Gambar 3.4: Contoh berkas "control" Paket Debian

Berkas **control** ini terdiri atas satu atau lebih paragraf kolom (*field*). Kolom-kolom tersebut adalah:

1) **Package** (wajib), kolom ini berisi nama paket yang dapat dimanipulasi atau dikontrol oleh perkakas manajer paket dan biasanya nama yang digunakan serupa tetapi tidak harus sama dengan *string* komponen pertama dalam nama berkas paket **Debian**.
2) **Source**, kolom ini berisi nama paket sumber yang kadang juga diikuti dengan nomor versi dalam tanda kurung.
3) **Version** (wajib), kolom ini berisi nomor versi termasuk nomor versi dari pengembang hulu (*upstream*) dan nomor revisi paket **Debian** (dalam komponen terakhir).
4) **Architecture** (wajib), kolom ini berisi nama tipe perangkat keras tertentu yang ditujukan sebagai target dari hasil kompilasi paket. Arsitektur mesin yang umum digunakan adalah **amd64**, **i386**, **armel**, **all** dan lain sebagainya. Catatan, jika kolom ini berisi atau bernilai "**all**", hal ini berarti paket merupakan paket *architecture independent* (tidak bergantung terhadap suatu arsitektur atau bisa dijalankan di berbagai arsitektur), sebagai contoh adalah paket yang berisi *shell script*, *script* **Python**, berkas konfigurasi dan dokumentasi.
5) Berbagai kolom yang menyatakan hubungan antara suatu paket dengan paket yang lain. Kolom-kolom tersebut adalah:
   a) **Depends**, kolom ini berisi daftar paket-paket yang diminta dan harus dipasang agar pemasangan paket dapat berhasil dan paket dapat berfungsi dengan baik.
   b) **Recommends**, kolom ini berisi daftar paket-paket yang tidak wajib dipasang walaupun memiliki tingkat kepentingan yang tinggi, karena akan meningkatkan fungsionalitas yang ditawarkan oleh paket walaupun sebenarnya tidak diperlukan dalam operasi.
   c) **Suggests**, kolom ini berisi daftar paket-paket yang dapat melengkapi dan meningkatkan kegunaan dari masing-masing paket, namun tidak akan terjadi masalah jika tidak kita pasang.
   d) **Enhances**, kolom ini berisi daftar paket-paket yang disarankan, tetapi dalam konteks yang berbeda. Keuntungannya adalah dimungkinkan untuk menambahkan saran tanpa harus memodifikasi paket yang bersangkutan. Jadi, semua *add-on*, *plug-in* dan ekstensi lain dari program nantinya dapat muncul dalam daftar saran yang berkaitan dengan perangkat lunak.
   e) **Pre-Depends**, kolom ini berisi daftar paket-paket yang dibutuhkan (dependensi), namun memiliki ketentuan yang lebih ketat. Paket ini harus dipasang secara lengkap dan dikonfigurasi dengan benar sebelum proses pemasangan paket yang menyatakankan sebagai "**Pre-Depends**" dimulai (yaitu sebelum dilakukannya eksekusi *script* **preinst** dan konfigurasi).
   f) **Conflicts**, kolom ini berisi daftar paket-paket paket yang tidak dapat dipasang secara bersamaan dengan paket yang lain.
   g) **Breaks**, kolom ini berisi daftar paket-paket yang jika dipasang dapat merusak paket yang lain (atau versi tertentu dari paket yang lain).
   h) **Provides**, kolom ini berisi nama paket virtual (**virtual package**) yang disediakan oleh paket.
   i) **Replaces**, kolom ini berisi daftar paket-paket yang berisi sejumlah berkas yang juga terdapat di paket yang lain, tetapi paket tersebut telah di-

berikan kewenangan untuk dapat menggantikannya (paket yang memiliki berkas yang sama tersebut).

j) **Built-Using**, kolom ini berisi daftar paket sumber tambahan yang digunakan pada waktu pembangunan paket biner.

k) **Built-For-Profiles**, kolom ini berisi daftar profil pembangunan yang digunakan pada pembangunan paket biner.

6) **Essential**, kolom ini berisi nilai *boolean* yang menunjukkan bahwa paket tersebut secara penuh (total) sangat diperlukan dalam kerja dan fungsi sistem. Jika kolom ini bernilai "*yes*", maka sistem manajemen paket akan menolak untuk menghapusnya (meningkatkan dan mengganti dengan paket yang memiliki fungsi yang sama masih dimungkinkan). Nilai lain yang mungkin dipakai adalah "*no*", yang berarti sama dengan paket yang tidak mencantumkan atau memiliki kolom ini sama sekali.

7) **Installed-Size**, kolom ini menunjukkan berapa banyak ruang disk yang akan dipakai setelah paket terpasang. Hal ini dimaksudkan untuk digunakan oleh manajer paket sehingga dapat menunjukkan apakah masih cukup ruang disk yang tersedia untuk pemasangan program.

8) **Section** (dianjurkan), kolom ini berisi nama kelompok paket yang didasarkan pada klasifikasi yang ditetapkan atas paket tersebut. Contoh **utils**, **base**, **net**, **mail**, **text**, **x11** dan lain sebagainya.

9) **Priority** (dianjurkan), kolom ini menunjukkan seberapa penting paket tersebut untuk dipasang oleh pengguna. Tingkatan prioritas ini ditetapkan oleh pemelihara distribusi sebagai bantuan untuk sistem manajemen paket. Berbagai nilai tingkatan prioritas tersebut adalah:

   a) **Required**, tingkatan ini menunjukkan bahwa paket tersebut diperlukan dalam kerja dan fungsi sistem (biasanya dapat juga diartikan bahwa fungsionalitas dari perkakas **dpkg** bergantung pada paket ini). Tingkatan ini meliputi semua perkakas yang sangat penting dalam perbaikan sistem yang rusak. Jika kita menghapus paket mungkin akan menyebabkan sistem menjadi rusak total.

   b) **Important**, tingkatan ini menunjukkan bahwa paket selalu dapat ditemukan pada sistem mirip **Unix**. Tanpa paket ini, paket yang lain tidak akan dapat berjalan dan dapat dipergunakan dengan baik.

   c) **Standard**, tingkatan ini menunjukkan bahwa paket merupakan standar dalam sistem **GNU/Linux**.

   d) **Optional**, tingkatan ini meliputi semua paket yang dapat kita pasang maupun tidak sesuai dengan keinginan kita dan tidak memiliki konflik atau pertentangan dengan paket dengan tingkatan prioritas yang lain.

   e) **Extra**. tingkatan ini meliputi semua paket yang dapat kita pasang namun memiliki konflik dengan tingkatan prioritas yang lain.

10) **Multi-Arch**, kolom ini berisi nilai yang menunjukkan tentang bagaimana paket harus bersikap pada sistem yang menerapkan pemasangan secara *multi-arch*. Berbagai nilai yang digunakan yaitu:

   a) **same**, nilai ini berarti paket dapat dipasang secara bersama dengan paket itu sendiri (yang berbeda arsitektur), tetapi tidak dapat memenuhi ketergantungan (dependensi) dari syuatu paket yang berbeda arsitektur.

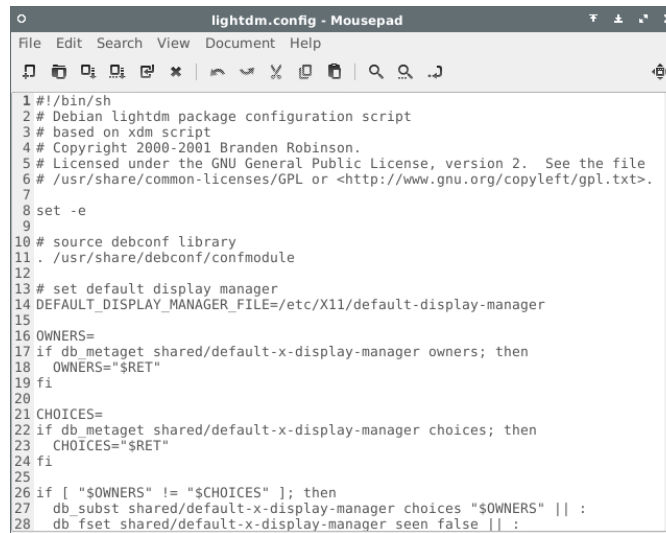
   b) **foreign**, nilai ini berarti paket tidak dapat dipasang bersama dengan paket itu sendiri (yang berbeda arsitektur), tetapi diperbolehkan untuk memenuhi ketergantungan yang tidak mensyaratkan suatu arsitektur tertentu dari paket yang berbeda arsitektur.

c) **allowed**, nilai ini memungkinkan adanya ketergantungan terbalik (*reserve dependencies*) yang menunjukkan di dalam kolom **Depens**-nya bahwa paket menerima paket yang berasa dari arsitektur yang berbeda dengan menentukan nama paket dengan atribut ":**any**", tetapi tidak untuk keadaan sebaliknya.
d) **no**, nilai ini merupakan nilai baku (*default*), yang juga berarti sama dengan jika kolom ini dihilangkan.

11) **Maintainer** (wajib), kolom ini berisi nama dan alamat lengkap surel (email) orang yang membuat paket, sebagai lawan dari pemilik perangkat lunak yang dipaketkan.
12) **Description** (wajib), kolom ini berisi gambaran atau penjelasan singkat tentang fitur-fitur yang dimiliki oleh paket. Kolom ini terdiri atas dua bagian, yaitu deskripsi singkat yang terdapat pada baris pertama kolom ini dan deskripsi panjang berisi penjelasan yang lebih rinci.
13) **Homepage**, kolom ini berisi halaman situs proyek hulu (*upstream*) perangkat lunak.

3.2.2.2. Script Pemelihara Paket

Script pemelihara paket (*package maintainer scripts*) adalah sejumlah *script* yang merupakan bagian dari paket yang akan dijalankan oleh sistem manajemen paket ketika pemasangan, peningkatan atau penghapusan paket. *Script-script* tersebut adalah:

1) **preinst**, yaitu *script* yang dijalankan sebelum paket **Debian** dibuka (*unpack*). Banyak *script* **preinst** menghentikan layanan (*service)* dari paket yang sedang ditingkatkan sampai proses pemasangan atau peningkatan selesai.
2) **postinst**, yaitu *script* yang dijalankan untuk melengkapi konfigurasi yang diperlukan oleh suatu paket segera sejak paket tersebut dibuka (*unpack*). Seringkali *script* **postinst** akan meminta pengguna untuk memasukkan suatu nilai dan atau memperingatkan pengguna jika pengguna menyetujui nilai baku (*default*), maka pengguna dapat kembali dan mengkonfigurasi ulang jika dibutuhkan. Banyak *script* **postinst** akan menjalankan perintah (*command*) yang diperlukan untuk menjalankan atau menjalankan ulang (*restart*) suatu layanan (*service*) segera setelah paket selesai dipasang atau ditingkatkan.
3) **prerm**, yaitu *script* yang dijalankan sebelum menghapus suatu paket atau berkas yang berhubungan dengan paket tersebut. *Script* ini biasanya akan menghentikan *service*, layanan atau proses yang berhubungan dengan paket tersebut.
4) **postrm**, yaitu *script* yang dijalankan segera setelah paket dihapus. Biasanya *script* ini akan mengubah tautan (*link*) atau berkas lain yang berhubungan dengan paket dan atau menghapus berkas yang dibuat oleh paket.

```
#!/bin/sh
# This script can be called in the following ways:
#
# Before the package is installed:
#       <new-preinst> install
#
# Before removed package is upgraded:
#       <new-preinst> install <old-version>
#
# Before the package is upgraded:
#       <new-preinst> upgrade <old-version>
#
#
# If postrm fails during upgrade or fails on failed upgrade:
#       <old-preinst> abort-upgrade <new-version>

set -e

# Handle upgrades from pre-conffile dpkg.cfg
upgrade_dpkg_non_conffile()
{
    if [ -r /etc/dpkg/dpkg.cfg ]; then
        dpkg_cfg_md5="535552ad5ee9145dbc7a34c264df4e59  /etc/dpkg/dpkg.cfg"
        if echo "$dpkg_cfg_md5" | md5sum -c >/dev/null 2>&1; then
            echo "Removing non-modified dpkg.cfg to be replaced by a conffile ..."
            rm -f /etc/dpkg/dpkg.cfg
```

Gambar 3.5: Contoh berkas Skrip Pemelihara Paket

Script-script pemeliharaan paket di atas dilengkapi dengan sebuah *script* **config** (berkas *script* ini menggunakan ekstensi nama berkas **.config**), yang disediakan oleh paket menggunakan perkakas **debconf** untuk memperoleh informasi dari pengguna sebagai konfigurasi. Selama proses pemasangan, *script* ini mendefinisikan secara rinci pertanyaan yang diajukan oleh perkakas **debconf**. Tanggapan akan dicatat di dalam basis data milik **debconf** sebagai referensi di masa mendatang. *Script* **preinst** dan **postinst** nantinya dapat menggunakan informasi ini untuk beroperasi sesuai dengan keinginan pengguna.

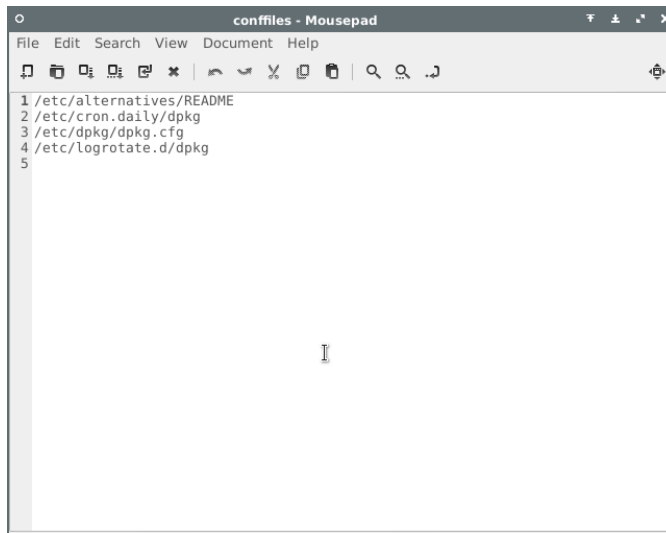
```
lightdm.config - Mousepad
File  Edit  Search  View  Document  Help

1 #!/bin/sh
2 # Debian lightdm package configuration script
3 # based on xdm script
4 # Copyright 2000-2001 Branden Robinson.
5 # Licensed under the GNU General Public License, version 2.  See the file
6 # /usr/share/common-licenses/GPL or <http://www.gnu.org/copyleft/gpl.txt>.
7
8 set -e
9
10 # source debconf library
11 . /usr/share/debconf/confmodule
12
13 # set default display manager
14 DEFAULT_DISPLAY_MANAGER_FILE=/etc/X11/default-display-manager
15
16 OWNERS=
17 if db_metaget shared/default-x-display-manager owners; then
18   OWNERS="$RET"
19 fi
20
21 CHOICES=
22 if db_metaget shared/default-x-display-manager choices; then
23   CHOICES="$RET"
24 fi
25
26 if [ "$OWNERS" != "$CHOICES" ]; then
27   db_subst shared/default-x-display-manager choices "$OWNERS" || :
28   db_fset shared/default-x-display-manager seen false || :
```

Gambar 3.6: Contoh berkas "config"

3.2.2.3. Berkas Checksums dan “conffiles”

Berkas *checksums* atau yang di dalam paket biasanya menggunakan nama berkas **md5sums** adalah berkas yang berisi **MD5 checksums** bagi semua berkas paket. Keuntungan utama dari adanya berkas *checksums* ini adalah dapat memungkinkan perkakas **dpkg –verify** untuk memeriksa apakah berkas-berkas paket pernah dilakukan modifikasi setelah pemasangannya. Sebagai catatan, jika berkas *checksums* tidak tersedia di dalam paket maka perkakas **dpkg** akan membuatnya secara dinamis pada saat proses pemasangan.

Sedangkan berkas *conffiles* adalah berkas yang berisi daftar berkas konfigurasi yang digunakan oleh paket (biasanya diletakkan dalam direktori ***/etc***). Berkas-berkas konfigurasi ini dapat dimodifikasi oleh administrator sistem dan sistem manajemen paket akan mencoba untuk mempertahankannya selama proses pembaharuan atau peningkatan paket. Maka dalam situasi yang demikian, sistem manajemen paket harus dapat bekerja dengan secerdas mungkin. Jika berkas konfigurasi standar antara kedua versi paket, maka hal tersebut bukanlah masalah. Namun, jika berkas tersebut telah berubah, maka sistem manajemen paket akan mencoba untuk memperbaharui berkas tersebut. Dua keadaan yang mungkin terjadi, yaitu:

1) administrator belum melakukan perubahan pada berkas konfigurasi, maka sistem manajemen paket secara otomatis memasang berkas konfigurasi yang baru, atau;
2) berkas konfigurasi telah dimodifikasi, maka sistem manajemen paket akan bertanya kepada administrator tentang berkas konfigurasi mana yang ingin digunakan (berkas konfigurasi lama yang telah dimodifikasi atau berkas konfigurasi baru yang disediakan oleh paket).

Gambar 3.7: Contoh berkas conffiles

Untuk membantu dalam keputusan ini, sistem manajemen paket akan menawarkan untuk menampilkan sebuah perbandingan yang dihasilkan menggunakan perkakas atau perintah "**diff**" yang akan menunjukkan perbedaan antara dua versi berkas konfigurasi tersebut. Jika pengguna memilih mempertahankan versi berkas konfigurasi yang lama, maka berkas konfigurasi yang baru akan disimpan dalam tempat yang sama dalam sebuah berkas dengan akhiran **.dpkg-dist**. Jika pengguna memilih versi yang baru, maka berkas konfigurasi yang lama akan dipertahankan dalam sebuah berkas dengan akhiran **.dpkg-old**. Tindakan lain yang tersedia adalah dengan menginterupsi proses sistem manajemen paket sehingga pengguna dapat mengedit berkas konfigurasi dan mencoba mengembalikan kembali modifikasi yang sesuai (yang sebelumnya diidentifikasi menggunakan perkakas **diff**).

3.2.2.4. Berkas "symbols" dan "shlibs"

Kedua berkas ini (berkas **symbols** dan **shlibs**) merupakan berkas yang menyediakan informasi tentang dependensi paket yang diperlukan untuk memastikan keberadaan *interface* atau antarmuka yang disediakan oleh suatu pustaka (*library*). Paket dengan berkas biner eksekutabel atau pustaka (*library*) yang berhubungan pada suatu pustaka bersama (*shared library*) akan menggunakan berkas tersebut untuk menentukan dependensi yang diperlukan ketika paket dibangun.

```
ppp.symbols - Mousepad
File  Edit  Search  View  Document  Help
 1 pppd.so.2.4.7 ppp #MINVER#
 2 #MISSING: 2.4.7-1+2# $global$@Base 2.4.7-1+2~
 3  ChallengeHash@Base 2.4.7-1+2~
 4  ChapMS2@Base 2.4.7-1+2~
 5  ChapMS@Base 2.4.7-1+2~
 6  DesDecrypt@Base 2.4.7-1+2~
 7  DesEncrypt@Base 2.4.7-1+2~
 8  DesSetkey@Base 2.4.7-1+2~
 9  GenerateAuthenticatorResponse@Base 2.4.7-1+2~
10  GetMask@Base 2.4.7-1+2~
11  MD4Final@Base 2.4.7-1+2~
12  MD4Init@Base 2.4.7-1+2~
13  MD4Print@Base 2.4.7-1+2~
14  MD4Update@Base 2.4.7-1+2~
15  MD5_Final@Base 2.4.7-1+2~
16  MD5_Init@Base 2.4.7-1+2~
17  MD5_Update@Base 2.4.7-1+2~
18  SHA1_Final@Base 2.4.7-1+2~
19  SHA1_Init@Base 2.4.7-1+2~
20  SHA1_Update@Base 2.4.7-1+2~
21  _IO_stdin_used@Base 2.4.7-1+2~
22  __libc_csu_fini@Base 2.4.7-1+2~
23  __libc_csu_init@Base 2.4.7-1+2~
24  _start@Base 2.4.7-1+2~
25  active_filter@Base 2.4.7-1+2~
26  add_fd@Base 2.4.7-1+2~
27  add_notifier@Base 2.4.7-1+2~
28  add_options@Base 2.4.7-1+2~
```

Gambar 3.8: Contoh berkas "sysmbols"

Berkas **symbols** menyediakan daftar *symbol* yang diekspor dari sebuah pustaka (*library*). Hal ini dapat membantu dalam menala atau menyetel dependensi dari pustaka (*library*) yang dihasilkan oleh perkakas **dpkg-shlibdeps** dan dalam pelacakan (sebagian) kompatibilitas biner. Hampir serupa dengan berkas **symbols**, berkas **shlibs** ini menyediakan pemetaan dari pustaka bersama (*shared libraryl)* terhadap kebutuhan atas informasi dependensi. Sistem **shlibs** ini merupakan alternatif dari sistem **symbols**, yang biasa digunakan untuk pustaka C ++ dan untuk keadaan lain yang pelacakan simbol individu terlalu sulit untuk dilakukan. Sistem **shlibs** ini lebih dahulu ada dibandingkan dengan sistem **symbols** dan oleh karena itu, lebih sering terlihat di paket yang lebih tua. Berkas **shlibs** ini juga diperlukan untuk **udeb**, yang tidak mendukung **symbols**.

3.2.2.5. Berkas “triggers”

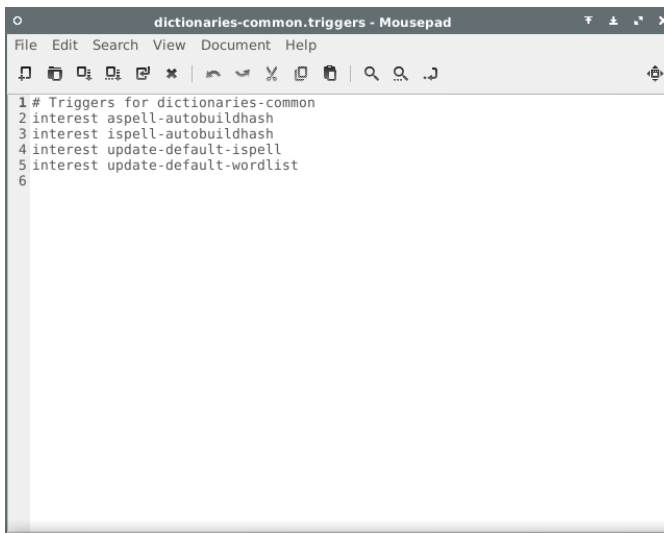
Berkas “**triggers**” merupakan berkas menyediakan informasi tentang hubungan suatu paket terhadap beberapa pemicu (*trigger*). Berkas ini berisi perintah, yang dibuat satu perintah per baris. Perintah kontrol *trigger* yang didukung saat ini adalah:

1) **interest**, perintah ini menentukan bahwa paket tertarik pada nama *trigger* tersebut. Format perintah yang digunakan adalah:

```
interest <nama_trigger>
interest-await <nama_trigger>
interest-noawait <nama_trigger>
```

2) **activate**, perintah ini mengatur aktivasi *trigger* tertentu pada saat terjadi perubahan status paket. *Trigger* akan diaktifkan pada awal dilakukannya kegiatan berikut, yaitu membuka (**unpack**), konfigurasi (**configure**), menghapus (**remove**), membersihkan (**purge**) dan menghapus konfigurasi (**deconfigure**). Format perintah yang digunakan adalah:

```
activate <nama_trigger>
```

```
activate-await <nama_trigger>
activate-noawait <nama_trigger>
```

Gambar 3.9: Contoh berkas triggers

3.2.3. Paket Biner Micro DEB (.udeb)

Paket **micro deb** merupakan sebuah jenis khusus dari paket **Debian** (paket biner). Paket **micro deb** ini adalah sebuah berkas paket **Debian** (paket biner) yang dipangkas untuk digunakan oleh program pemasang sistem **Debian** (**debian-installer**) dan tidak dapat dipasang dalam sistem yang normal. Secara teknis, paket dalam format ini sangat mirip dengan paket-paket **Debian** pada umumnya. Hal-hal utama yang membedakan keduanya meliputi:

1) dalam paket **micro deb**, banyak persyaratan kebijakan (*policy*) yang dibebaskan. Sebagai contoh, sebuah berkas **micro deb** tidak berisi berkas **changelog**, lisensi, **manpage** atau **md5sum**. Alasannya adalah untuk meminimalkan ukuran sehingga dapat menghemat kebutuhan memori saat proses pemasangan sistem operasi.
2) paket **micro deb** tidak benar-benar dimaksudkan untuk dapat dihapus atau dinaikkan (*upgrade*).

3.3. Paket Sumber Debian

Paket sumber **Debian** ini merupakan bentuk atau format standar dari paket atau kumpulan berkas kode sumber perangkat lunak yang digunakan untuk menghasilkan paket biner dalam **Sistem Manajemen Paket Debian**. Tujuannya adalah untuk membantu mempermudah dan mengotomatisasi pembangunan paket biner. Meskipun kata “paket” ini memiliki konotasi atau arti sebagai satu berkas tunggal yang mengemas berbagai macam isi, paket sumber **Debian** ini sebenarnya terdiri atas dua atau tiga buah berkas. Berkas-berkas tersebut yaitu:

1) berkas **.dsc** (**Debian Source Control**).
2) berkas **.orig.tar.gz** (atau bisa juga menggunakan format kompresi yang lain yang didukung seperti **.tar.bz2**, **.tar.xz** dan **.tar.lzma**), berkas ini merupakan

sebuah berkas arsip yang berisi kode sumber yang disediakan oleh pengembang aslinya (pengembang hulu atau *upstream*). Pemelihara (*maintainer*) paket **Debian** diminta untuk tidak memodifikasi berkas arsip ini agar dapat mempermudah dalam memeriksa keaslian dan keutuhan berkas (melalui perbandingan sederhana dengan sebuah *checksum*) serta untuk memenuhi keinginan dari beberapa penulis.

3) berkas **.debian.tar.gz** (atau bisa juga menggunakan format kompresi yang lain yang didukung seperti **.tar.bz2**, **.tar.xz** dan **.tar.lzma**), berkas ini merupakan berkas arsip yang berisi semua perubahan yang dibuat oleh pemelihara (*maintainer*) paket **Debian**, khususnya tambahan berupa sebuah direktori dengan nama "**debian**" yang berisi instruksi (perintah) untuk menjalankan pembangunan paket (biner) **Debian**.

Paket sumber **Debian** ini terdiri atas dua macam tipe, yaitu:

1) paket *native*, yaitu paket yang berisi perangkat lunak yang secara khusus dikembangkan melalui **Proyek Debian**, seperti **dpkg**, **APT** dan lain sebagainya. Dalam tipe paket ini, paket sumber terdiri atas dua berkas, yaitu berkas **Debian Source Control** (**.dsc**) dan berkas arsip kode sumber perangkat lunak biasanya menggunakan ekstensi nama berkas **.tar.gz**, **.tar.xz**, atau **.tar.bz2** (tanpa ada tambahan nama **.orig**).
2) paket *non-native*, yaitu paket yang berisi perangkat lunak yang dikembangkan di luar dari **Proyek Debian**.

Format paket sumber **Debian** saat ini telah mencapai versi 3.0, yang diterapkan pertama kali pada **dpkg 1.14.17** dan mulai dipergunakan secara resmi pada sebuah rilis stabil dalam rilis **Debian 5.0.4**.

3.3.1. Berkas Debian Source Control (.dsc)

Berkas **Debian Source Control** (**.dsc**) adalah berkas teks singkat yang menjelaskan tentang paket sumber dan dan menunjukkan berkas-berkas lain yang menjadi bagian dari paket sumber tersebut. Seperti halnya berkas **control** yang terdapat pada paket biner **Debian**, berkas **.dsc** ini terdiri atas sejumlah *field* (kolom). Kolom-kolom tersebut adalah:

1) **Format** (wajib), kolom ini berisi versi format paket sumber yang digunakan. Format paket sumber yang saat ini didukung oleh **dpkg** adalah **1.0**, **2.0**, **3.0** (**native**), **3.0** (**quilt**), **3.0** (**git**), **3.0** (**bzr**) dan **3.0** (**custom**);
2) **Source** (wajib), kolom ini berisi nama dari paket sumber;
3) **Binary**, kolom ini berisi daftar nama paket biner yang dapat dihasilkan dari paket sumber. Kolom ini saat ini telah digantikan oleh kolom **Package-List**, yang dapat memberikan informasi yang cukup tentang paket biner apa yang dihasilkan dalam berbagai arsitektur yang didukung, profil pembangunan paket dan pembatasan lain yang berkaitan;
4) **Architecture** (dianjurkan), kolom ini berisi daftar kolom ini berisi nama tipe perangkat keras tertentu yang ditujukan sebagai target dari hasil kompilasi paket. Catatan, jika kolom ini berisi atau bernilai "**all**", hal ini berarti paket merupakan paket *architecture independent* (tidak bergantung terhadap suatu arsitektur atau bisa dijalankan di berbagai arsitektur). Sedangkan jika kolom bernilai "**any**", berarti paket merupakan paket *architecture dependent* (bergantung pada arsitektur tertentu atau hanya bisa dijalankan di arsitektur tertentu);

5) **Version** (wajib), kolom ini berisi nomor versi paket. Biasanya nomor versi yang digunakan adalah nomor versi asli paket yang berasal dari pengembang aslinya. Namun, juga kadang mengikutsertakan nomor revisi **Debian** (untuk paket *non-native*);
6) **Origin**, kolom ini berisi nama distribusi asal dari paket;
7) **Maintainer** (dianjurkan), kolom ini berisi nama dan alamat lengkap surel (*email*) orang yang membuat paket, sebagai lawan dari pemilik perangkat lunak yang dipaketkan;
8) **Uploaders**, kolom ini berisi daftar nama dan alamat surel dari para pemelihara bersama paket (*co-maintainer*);
9) **Homepage**, kolom ini berisi alamat URL halaman situs proyek hulu (*upstream*) perangkat lunak;
10) **Standards- Version** (dianjurkan), kolom ini berisi rangakaian nomor versi dari standar kebijakan distribusi yang digunakan atau diikuti oleh paket. Penjelasan lebih lengkap tentang hal ini, dapat kita baca di alamat web https://www.debian.org/doc/packaging-manuals/upgrading-checklist.txt;
11) **Vcs-Browser**, kolom ini berisi alamat URL dari antarmuka web yang digunakan untuk menelusuri repositori sistem pengaturan versi perangkat lunak atau biasa dikenal dengan sebutan **Vcs** (*Version Control System*);
12) **Vcs-Arch**, **Vcs-Bzr**, **Vcs-Cvs**, **Vcs-Darcs**, **Vcs-Git**, **Vcs-Hg**, **Vcs-Mtn** dan **Vcs-Svn**, kolom-kolom ini berisi alamat URL dari repositosi sistem pengaturan versi yang digunakan untuk memelihara paket. Sistem pengaturan versi yang saat ini didukung adalah **Arch**, **Bzr** (**Bazaar**), **Cvs**, **Darcs**, **Git**, **Hg** (**Mercurial**), **Mtn** (**Monotone**) dan **Svn** (**Subversion**). Biasanya kolom ini akan merujuk pada versi terbaru dari paket, seperti cabang utama (*main*) atau *trunk* dari perangkat lunak tersebut;
13) **Testsuite**, kolom ini berisi daftar nama sekumpulan alat atau perintah tes (pengujian) tertentu yang terdapat pada berkas paket sumber;
14) **Build-Depends**, kolom ini berisi daftar nama paket yang dibutuhkan (telah terpasang dan terkonfigurasi) untuk dapat membangun dari paket sumber (baik ketika membangun paket biner *architecture dependent* maupun paket *architecture independent* serta paket sumber);
15) **Build-Depends-Arch**, kolom ini berisi daftar nama paket yang hanya dibutuhkan ketika membangun paket *architecture dependent*;
16) **Build-Depends-Indep**, kolom ini berisi daftar nama paket yang hanya dibutuhkan ketika membangun paket *architecture independent*;
17) **Build-Conflicts**, kolom ini berisi daftar paket yang tidak boleh dipasang ketika paket sedang dibangun, misalnya karena dapat mengganggu sistem pembangunan paket;
18) **Build-Conflicts-Arch**, kolom ini berisi daftar paket yang tidak boleh dipasang hanya ketika membangun paket *architecture dependent*;
19) **Build-Conflicts-Indep**, kolom ini berisi daftar paket yang tidak boleh dipasang hanya ketika membangun paket *architecture independent*;
20) **Package-List**, kolom yang dapat dibuat banyak baris ini berisi daftar paket biner yang dibuat menggunakan paket sumber ini. Nilainya berisi nama paket biner, tipe paket biner (**.deb** atau **.udeb**), bagian (*section*) dan prioritas (*priority*) yang harus sesuai dengan kolom yang sama yang ada di berkas **control** paket biner serta daftar nilai kunci (*key-value-list*). Format penulisannya adalah "**<nama paket> <tipe paket> <section> <priority> <daftar nilai kunci>**". Kunci-kunci opsional yang dikenali saat ini adalah:

a) **arch**, berisi batasan arsitektur yang berasal dari kolom **Architecture** paket biner.
b) **essential**, jika paket biner merupakan paket yang esensial (sangat penting), kunci ini akan berisi nilai yang sama dengan yang ada di kolom **Essensial** paket biner, yaitu bernilai **yes**.

21) **Checksums-Sha1** (wajib), kolom ini berisi daftar berkas paket sumber (selain berkas **.dsc**) dan informasi tentang tiap berkas, yaitu berupa **SHA-256 Checksums** berkas, ukuran dan nama berkas.
22) **Checksums-Sha256** (wajib), kolom ini berisi daftar berkas paket sumber (selain berkas **.dsc**) dan informasi tentang tiap berkas, yaitu berupa **SHA-256 Checksums** berkas, ukuran dan nama berkas.
23) **Files** (wajib), kolom ini berisi daftar berkas paket sumber (selain berkas **.dsc**) dan informasi tentang tiap berkas, yaitu berupa **MD5 Checksums** berkas, ukuran dan nama berkas.

```
dpkg_1.18.7.dsc - Mousepad
File  Edit  Search  View  Document  Help

1 -----BEGIN PGP SIGNED MESSAGE-----
2 Hash: SHA512
3
4 Format: 3.0 (native)
5 Source: dpkg
6 Binary: libdpkg-dev, dpkg, dpkg-dev, libdpkg-perl, dselect
7 Architecture: any all
8 Version: 1.18.7
9 Origin: debian
10 Maintainer: Dpkg Developers <debian-dpkg@lists.debian.org>
11 Uploaders: Guillem Jover <guillem@debian.org>
12 Homepage: https://wiki.debian.org/Teams/Dpkg
13 Standards-Version: 3.9.8
14 Vcs-Browser: https://anonscm.debian.org/cgit/dpkg/dpkg.git
15 Vcs-Git: https://anonscm.debian.org/git/dpkg/dpkg.git
16 Build-Depends: dpkg-dev (>= 1.17.14), debhelper (>= 9.20141010), pkg-config, gettext (>= 0.19), po4a (>= 0.41), zlib1g-dev, libbz2-dev, liblzma-dev, libselinux1-dev
   (>= 1.28-4) [linux-any], libncursesw5-dev, libio-string-perl <!nocheck>
17 Package-List:
18  dpkg deb admin required arch=any essential=yes
19  dpkg-dev deb utils optional arch=all
20  dselect deb admin optional arch=any
21  libdpkg-dev deb libdevel optional arch=any
22  libdpkg-perl deb perl optional arch=all
23 Checksums-Sha1:
24  dd223bc6f70f43075cc8b7a3ec4925500ff6be5e 4617284 dpkg_1.18.7.tar.xz
25 Checksums-Sha256:
26  ace36d3a6dc750a42baf797f9e75ec580a21f92bb9ff96b482100755d6d9b87b 4617284 dpkg_1.18.7.tar.xz
27 Files:
28  073dbf2129a54b0fc627464bf8af4a1b 4617284 dpkg_1.18.7.tar.xz
29
30 -----BEGIN PGP SIGNATURE-----
31 Version: GnuPG v2
32
33 iQIcBAEBCgAGBQJXL/+FAAoJELlyvz6krlejduAP/iuFUMZUm+4PhDXTE2GdSxfG
34 egIhtTaeZ9TveQQ4vHKJEyDRxKG7harlg/EZrey6zCWEH7FWO+lnUaTcqaNk/fYZ
35 3zWn3QUHHYpDzZsACyUu3L1t50DXQa2EkJKB46mKYQBVERtQdvtWiae0lC2L0hO1
36 O7APld1WNad5nO3umcdV4VHciNd98oPcFee6WpUvBGfK2UjCPjKlffKp28ETC3Eh
37 3Bfns4dKyLvKMDQdUAsfdkxw8beB19k4LR02p29GcPbKU8ahHwW2zCJNrLqzteg7
38 wDruLIk+X8h7VaMCEHMeVNpGcgHryrpyVqvQbqOZm+Jy6oa/lj0ifJcqZRonITXi
39 4wOQ2nWOVPTzvwNX6NxM54ZBxjZH4MVCwnu4ATnmnJ6WL6OStOhs+0TFkthYSz9L
40 z/QAMTdU/RwyK2p+89PoHCvABApTK3uAapnwvKpec7LTHSAdYRj1fSA6Qf9Bwh/q
41 qRjX+i6wwSFGDmX1Xq5plCRpPKyT81mcsPlKVmKVKWFE0PzsuG2ntNW/FV3qKiKL
42 k8vqeUJcNAEeVg3Ou1nyz3gd6JIh2VEnTygwjkqnmdRjXOXHVGmuqhgnFWFaBnNH
```

Gambar 3.10: Contoh berkas Debian Source Control

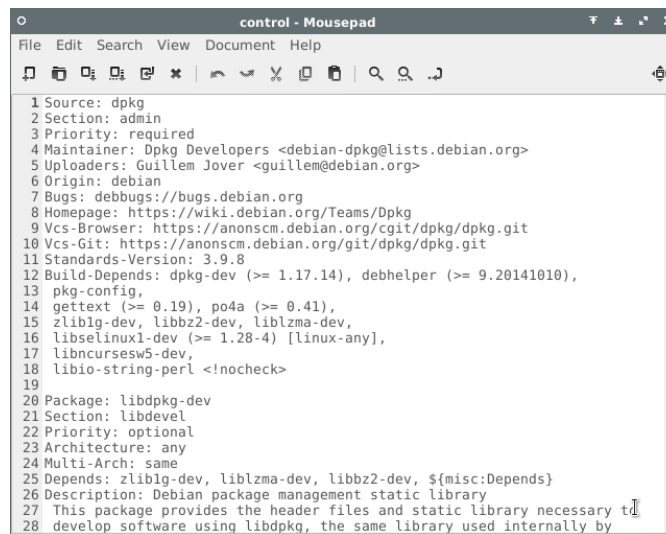
Data **control** pada berkas **.dsc** in harus tertutup dalam tanda tangan (*signature*) **OpenPGP ASCI Armored**, sebagaimana ditentukan dalam **RFC 4880**.

3.3.2. Berkas-berkas Wajib dalam direktori “debian”

Sebagian besar perubahan dalam paket sumber **Debian** ditempatkan dalam direktori **debian** ini yang dapat kita temukan di dalam berkas arsip **.debian.tar.gz** untuk paket sumber *non-native* dan dalam berkas arsip **.tar.gz** untuk paket sumber *native*. Di dalam direktori ini terdapat sejumlah berkas, yang di antaranya merupakan berkas-berkas yang sangat penting yang dibutuhkan (diharuskan ada) bagi semua paket, yaitu berkas **control**, **changelog**, **copyright** dan **rules**.

3.3.2.1. Berkas “control”

Berkas **control** ini adalah berkas yang berisi berbagai nilai yang digunakan oleh perkakas manajemen paket dalam melaksanakan pengelolaan paket. Ber-

kas **control** ini serupa dengan berkas **control** yang ada di dalam paket biner **Debian** dan berkas **Debian Source Control**. Berkas ini berisi sejumlah kolom (*field*) yang terbagi atas dua bagian yaitu:

1) Bagian kolom sumber (*source field*), bagian ini berisi segala informasi tentang paket sumber secara umum. Kolom-kolom yang termasuk dalam bagian ini yaitu:
   a) **Source** (wajib), kolom ini berisi nama dari paket sumber.
   b) **Maintainer** (dianjurkan), kolom ini berisi nama dan alamat lengkap surel (*email*) orang yang membuat paket, sebagai lawan dari pemilik perangkat lunak yang dipaketkan.
   c) **Uploaders**, kolom ini berisi daftar nama dan alamat surel dari para pemelihara bersama paket (*co-maintainer*).
   d) **Standards- Version** (dianjurkan), kolom ini berisi rangakaian nomor versi dari standar kebijakan distribusi yang digunakan atau diikuti oleh paket. Penjelasan lebih lengkap tentang hal ini, dapat kita baca di alamat web https://www.debian.org/doc/packaging-manuals/upgrading-checklist.txt.
   e) **Homepage**, kolom ini berisi alamat URL halaman situs proyek hulu (*upstream*) perangkat lunak.
   f) **Bugs**, kolom ini berisi alamat URL sistem pelacakan kutu (*bug*) yang digunakan untuk paket.
   g) **Vcs-Arch**, **Vcs-Bzr**, **Vcs-Cvs**, **Vcs-Darcs**, **Vcs-Git**, **Vcs-Hg**, **Vcs-Mtn** dan **Vcs-Svn**, kolom-kolom ini berisi alamat URL dari repositosi sistem pengaturan versi yang digunakan untuk memelihara paket. Sistem pengaturan versi yang saat ini didukung adalah **Arch**, **Bzr** (**Bazaar**), **Cvs**, **Darcs**, **Git**, **Hg** (**Mercurial**), **Mtn** (**Monotone**) dan **Svn** (**Subversion**). Biasanya kolom ini akan merujuk pada versi terbaru dari paket, seperti cabang utama (*main*) atau *trunk* dari perangkat lunak tersebut.
   h) **Vcs-Browser**, kolom ini berisi alamat URL dari antarmuka web yang digunakan untuk menelusuri repositori sistem pengaturan versi perangkat lunak atau biasa dikenal dengan sebutan **Vcs** (*Version Control System*).
   i) **Origin**, kolom ini berisi nama distribusi asal dari paket.
   j) **Section**, kolom ini berisi nama kelompok paket yang didasarkan pada klasifikasi yang ditetapkan atas paket tersebut. Contoh **utils**, **base**, **net**, **mail**, **text**, **x11** dan lain sebagainya.
   k) **Priority**, kolom ini menunjukkan seberapa penting paket tersebut untuk dipasang oleh pengguna. Tingkatan prioritas ini ditetapkan oleh pemelihara distribusi sebagai bantuan untuk sistem manajemen paket. Berbagai nilai tingkatan prioritas tersebut adalah:
      i) **Required**, tingkatan ini menunjukkan bahwa paket tersebut diperlukan dalam kerja dan fungsi sistem (biasanya dapat juga diartikan bahwa fungsionalitas dari perkakas **dpkg** bergantung pada paket ini). Tingkatan ini meliputi semua perkakas yang sangat penting dalam perbaikan sistem yang rusak. Jika kita menghapus paket mungkin akan menyebabkan sistem menjadi rusak total .
      ii) **Important**, tingkatan ini menunjukkan bahwa paket selalu dapat ditemukan pada sistem mirip **Unix**. Tanpa paket ini, paket yang lain tidak akan dapat berjalan dan dapat dipergunakan dengan baik.
      iii) **Standard**, tingkatan ini menunjukkan bahwa paket merupakan standar dalam sistem **GNU/Linux**.

iv) **Optional**, tingkatan ini meliputi semua paket yang dapat kita pasang maupun tidak sesuai dengan keinginan kita dan tidak memiliki konflik atau pertentangan dengan paket dengan tingkatan prioritas yang lain.

v) **Extra**. tingkatan ini meliputi semua paket yang dapat kita pasang namun memiliki konflik dengan tingkatan prioritas yang lain.

l) **Build-Depends**, kolom ini berisi daftar nama paket yang dibutuhkan (telah terpasang dan terkonfigurasi) untuk dapat membangun dari paket sumber (baik ketika membangun paket biner *architecture dependent* maupun paket *architecture independent* serta paket sumber).

m) **Build-Depends-Arch**, kolom ini berisi daftar nama paket yang hanya dibutuhkan ketika membangun paket *architecture dependent*.

n) **Build-Depends-Indep**, kolom ini berisi daftar nama paket yang hanya dibutuhkan ketika membangun paket *architecture independent*.

o) **Build-Conflicts**, kolom ini berisi daftar paket yang tidak boleh dipasang ketika paket sedang dibangun, misalnya karena dapat mengganggu sistem pembangunan paket.

p) **Build-Conflicts-Arch**, kolom ini berisi daftar paket yang tidak boleh dipasang hanya ketika membangun paket *architecture dependent*.

q) **Build-Conflicts-Indep**, kolom ini berisi daftar paket yang tidak boleh dipasang hanya ketika membangun paket *architecture independent*.

2) Bagian kolom biner (*binary field*), bagian ini berisi informasi tentang paket-paket biner yang akan dihasilkan. Kolom-kolom yang termasuk dalam bagian ini yaitu:

a) **Package** (wajib), kolom ini berisi nama paket biner.

b) **Architecture** (wajib), kolom ini berisi nama tipe perangkat keras tertentu yang dapat menjalankan paket. Nilai “**any**” digunakan untuk paket yang dapat berjalan pada semua arsitektur. Nilai “**all**” digunakan untuk paket yang merupakan paket *architecture independent* seperti *shell script*, *script* **Python**, berkas konfigurasi dan dokumentasi. Sedangkan untuk membatasi paket pada beberapa arsitektur tertentu, kita dapat menggunakan nama arsitektur secara khusus.

c) **Build-Profiles**, kolom ini berisi formula atau rumus pembatasan (*restriction formula*) yang menentukan kondisi apakah paket akan dibangun atau tidk.

d) **Package-Type**, kolom ini berisi tipe paket. Nilai **udeb**, berarti paket merupakan paket yang digunakan oleh program *installer* **Debian**, sedangkan nilai **deb** adalah nilai baku (*default*).

e) **Essential**, kolom ini berisi nilai *boolean* yang menunjukkan bahwa paket tersebut secara penuh (total) sangat diperlukan dalam kerja dan fungsi sistem. Jika kolom ini bernilai “*yes*”, maka sistem manajemen paket akan menolak untuk menghapusnya (meningkatkan dan mengganti dengan paket yang memiliki fungsi yang sama masih dimungkinkan). Nilai lain yang mungkin dipakai adalah “*no*”, yang berarti sama dengan paket yang tidak mencantumkan atau memiliki kolom ini sama sekali.

f) **Multi-Arch**, kolom ini berisi nilai yang menunjukkan tentang bagaimana paket harus bersikap pada sistem yang menerapkan pemasangan secara *multi-arch*. Berbagai nilai yang digunakan yaitu:

i) **same**, nilai ini berarti paket dapat dipasang secara bersama dengan paket itu sendiri (yang berbeda arsitektur), tetapi tidak dapat meme-

nuhi ketergantungan (dependensi) dari syuatu paket yang berbeda arsitektur.

ii) **foreign**, nilai ini berarti paket tidak dapat dipasang bersama dengan paket itu sendiri (yang berbeda arsitektur), tetapi diperbolehkan untuk memenuhi ketergantungan yang tidak mensyaratkan suatu arsitektur tertentu dari paket yang berbeda arsitektur.

iii) **allowed**, nilai ini memungkinkan adanya ketergantungan terbalik (*reserve dependencies*) yang menunjukkan di dalam kolom **Depens**-nya bahwa paket menerima paket yang berasa dari arsitektur yang berbeda dengan menentukan nama paket dengan atribut ":**any**", tetapi tidak untuk keadaan sebaliknya.

iv) **no**, nilai ini merupakan nilai baku (*default*), yang juga berarti sama dengan jika kolom ini dihilangkan.

g) **Description** (dianjurkan), kolom ini berisi gambaran atau penjelasan singkat tentang fitur-fitur yang dimiliki oleh paket.

h) Kolom-kolom **Priority**, **Section** dan **Hompage** juga dapat dicantumkan dalam bagian kolom biner ini untuk menimpa nilai global yang berasal dari paket sumber (nilai atau isi untuk kolom yang sama).

i) Berbagai kolom yang menyatakan hubungan antara suatu paket dengan paket yang lain. Kolom-kolom tersebut adalah:

  i) **Depends**, kolom ini berisi daftar paket-paket yang diminta dan harus dipasang agar pemasangan paket dapat berhasil dan paket dapat berfungsi dengan baik.

  ii) **Recommends**, kolom ini berisi daftar paket-paket yang tidak wajib dipasang walaupun memiliki tingkat kepentingan yang tinggi, karena akan meningkatkan fungsionalitas yang ditawarkan oleh paket walaupun sebenarnya tidak diperlukan dalam operasi.

  iii) **Suggests**, kolom ini berisi daftar paket-paket yang dapat melengkapi dan meningkatkan kegunaan dari masing-masing paket, namun tidak akan terjadi masalah jika tidak kita pasang.

  iv) **Enhances**, kolom ini berisi daftar paket-paket yang disarankan, tetapi dalam konteks yang berbeda. Keuntungannya adalah dimungkinkan untuk menambahkan saran tanpa harus memodifikasi paket yang bersangkutan. Jadi, semua *add-on*, *plug-in* dan ekstensi lain dari program nantinya dapat muncul dalam daftar saran yang berkaitan dengan perangkat lunak.

  v) **Pre-Depends**, kolom ini berisi daftar paket-paket yang dibutuhkan (dependensi), namun memiliki ketentuan yang lebih ketat. Paket ini harus dipasang secara lengkap dan dikonfigurasi dengan benar sebelum proses pemasangan paket yang menyatakan sebagai "**Pre-Depends**" dimulai (yaitu sebelum dilakukannya eksekusi *script* **preinst** dan konfigurasi).

  vi) **Conflicts**, kolom ini berisi daftar paket-paket paket yang tidak dapat dipasang secara bersamaan dengan paket yang lain.

  vii) **Breaks**, kolom ini berisi daftar paket-paket yang jika dipasang dapat merusak paket yang lain (atau versi tertentu dari paket yang lain).

  viii) **Provides**, kolom ini berisi nama paket virtual (**virtual package**) yang disediakan oleh paket.

  ix) **Replaces**, kolom ini berisi daftar paket-paket yang berisi sejumlah berkas yang juga terdapat di paket yang lain, tetapi paket tersebut

telah diberikan kewenangan untuk dapat menggantikannya (paket yang memiliki berkas yang sama tersebut).

```
control - Mousepad
File  Edit  Search  View  Document  Help
 1 Source: dpkg
 2 Section: admin
 3 Priority: required
 4 Maintainer: Dpkg Developers <debian-dpkg@lists.debian.org>
 5 Uploaders: Guillem Jover <guillem@debian.org>
 6 Origin: debian
 7 Bugs: debbugs://bugs.debian.org
 8 Homepage: https://wiki.debian.org/Teams/Dpkg
 9 Vcs-Browser: https://anonscm.debian.org/cgit/dpkg/dpkg.git
10 Vcs-Git: https://anonscm.debian.org/git/dpkg/dpkg.git
11 Standards-Version: 3.9.8
12 Build-Depends: dpkg-dev (>= 1.17.14), debhelper (>= 9.20141010),
13  pkg-config,
14  gettext (>= 0.19), po4a (>= 0.41),
15  zlib1g-dev, libbz2-dev, liblzma-dev,
16  libselinux1-dev (>= 1.28-4) [linux-any],
17  libncursesw5-dev,
18  libio-string-perl <!nocheck>
19
20 Package: libdpkg-dev
21 Section: libdevel
22 Priority: optional
23 Architecture: any
24 Multi-Arch: same
25 Depends: zlib1g-dev, liblzma-dev, libbz2-dev, ${misc:Depends}
26 Description: Debian package management static library
27  This package provides the header files and static library necessary to
28  develop software using libdpkg, the same library used internally by
```

Gambar 3.11: Contoh berkas control dalam paket sumber

3.3.2.2. Berkas “copyright”

Berkas **copyright** ini berisi informasi tentang hak cipta dan lisensi dari kode sumber hulu (*upstream*). Berkas ini tidak diperbolehkan dalam bentuk berkas terkompresi maupun dalam bentuk tautan simbolis (*symbolic link*) serta harus dibuat menggunakan standar pengkodean **UTF-8**. Bagi paket yang didistribusikan di bawah lisensi **Apache** (versi 2.0), lisensi **Artistic**, **GNU GPL** (versi 1, 2, atau 3), **GNU LGPL** (versi 2, 2.1, atau 3) dan **GNU FDL** (versi 1.2 atau 1.3), harus dibuat rujukan ke berkas yang berada di dalam direktori **/usr/share/common-licences/** (yang tersedia secara baku dalam sistem **Debian**, termasuk sistem turunannya), daripada mengutip teks lisensinya secara lengkap dalam berkas **copyright**. Bagi paket yang menggunakan lisensi yang lain maka teks lisensi paket tersebut harus dicantumkan secara lengkap dalam berkas **copyright**. Setelah paket terpasang, berkas **copyright** ini akan disimpan sebagai berkas **/usr/share/doc/nama-paket/copyright**.

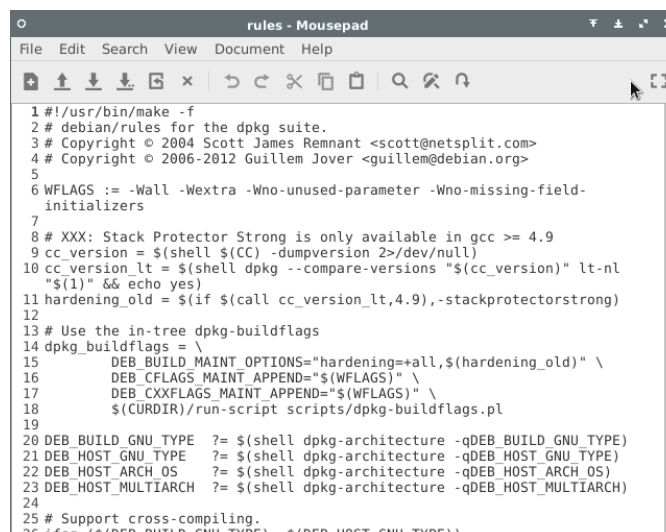
```
copyright - Mousepad
File Edit Search View Document Help

1 Apt is copyright 1997, 1998, 1999 Jason Gunthorpe and others.
2 Apt is currently developed by APT Development Team <deity@lists.debian.org>.
3
4 License: GPLv2+
5
6     This program is free software; you can redistribute it and/or modify
7     it under the terms of the GNU General Public License as published by
8     the Free Software Foundation; either version 2 of the License, or
9     (at your option) any later version.
10
11    This program is distributed in the hope that it will be useful,
12    but WITHOUT ANY WARRANTY; without even the implied warranty of
13    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
14    GNU General Public License for more details.
15
16    You should have received a copy of the GNU General Public License
17    along with this program; if not, write to the Free Software
18    Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301,
  USA.
19
20 See /usr/share/common-licenses/GPL-2, or
21 <http://www.gnu.org/copyleft/gpl.txt> for the terms of the latest version
22 of the GNU General Public License.
23
24
```

Gambar 3.12: Contoh berkas copyright

3.3.2.3. Berkas "changelog"

Berkas **changelog** ini berisi catatan perubahan yang dilakukan terhadap paket **Debian** yang dibandingkan dengan paket hulu (*upstream*), termasuk perubahan-perubahan lain dan pembaharuan untuk paket. Berkas ini juga harus dibuat menggunakan standar pengkodean **UTF-8**.

```
changelog - Mousepad
File Edit Search View Document Help

 1 apt (1.2.12) unstable; urgency=medium
 2
 3   [ Patrick Cable ]
 4   * refactored no_proxy code to work regardless of where https proxy is
   set
 5
 6   [ James McCoy ]
 7   * deb822: Restore support for <multivalue>-{Add,Remove}
 8
 9   [ David Kalnischkies ]
10   * don't show NO_PUBKEY warning if repo is signed by another key
11     (Closes: 618445)
12   * allow redirection for items without a space in the desc again
13   * don't sent uninstallable rc-only versions via EDSP
14   * respect user pinning in M-A:same version (un)screwing
15
16   [ Julian Andres Klode ]
17   * update: Run Post-Invoke-Success if not all sources failed
18   * debian/gbp.conf: Set debian-branch to 1.2.y
19
20   [ Frans Spiesschaert ]
21   * Dutch program translation update (Closes: 823976)
22
23  -- Julian Andres Klode <jak@debian.org>  Wed, 11 May 2016 10:56:53
   +0200
24
25 apt (1.2.11) unstable; urgency=medium
26
```

Gambar 3.13: Contoh berkas changelog

3.3.2.4. Berkas "rules"

Berkas **rules** ini merupakan berkas yang berisi *script* yang mengatur bagaimana paket biner dibangun. Berkas ini sebenarnya adalah bentuk lain dari berkas **Makefile**, namun berbeda dengan berkas **Makefile** yang berasal dari kode sumber hulu (*upstream*). Tidak seperti berkas-berkas lain yang berada di dalam

direktori **debian**, berkas **rules** ini ditandai sebagai berkas eksekutabel (diberikan hak akses eksekutabel).

Setiap berkas **rules**, sebagaimana berkas **Makefile** yang lain, terdiri atas beberapa aturan (*rule*) yang mendefinisikan tentang suatu target dan bagaimana target tersebut dilakukan. Target-target yang didukung di antaranya adalah:

1) **clean** (wajib), target ini digunakan untuk membersihkan semua berkas hasil kompilasi, berkas yang dihasilkan (*generated files*) dan berkas yang tidak berguna yang ada di dalam *build-tree*;
2) **build** (wajib), target ini digunakan untuk membangun kode sumber menjadi program terkompilasi dan dokumen terformat dalam *build-tree*;
3) **build-arch** (wajib), target ini digunakan membangun kode sumber menjadi program terkompilasi yang *architecture dependent* dalam *build-tree*;
4) **build-indep** (wajib), target ini digunakan untuk membangun kode sumber menjadi dokumen terformat yang *architecture independent* dalam *build-tree*;
5) **install** (opsional), target ini digunakan untuk memasang berkas ke dalam sebuah pohon berkas untuk setiap paket biner di bawah direktori **debian**. Jika ditetapkan, maka target-target **binary** akan bergantung pada target ini ;
6) **binary** (wajib), target ini digunakan untuk membuat semua paket biner (gabungan dari target **binary-arch** dan **binary-indep**);
7) **binary-arch** (wajib), target ini digunakan untuk membuat paket biner yang *architecture dependent* dalam direktori induk;
8) **binary-indep** (wajib), target ini digunakan untuk membuat paket biner yang *architecture independent* dalam direktori induk; dan
9) **get-orig-source** (opsional), target ini digunakan untuk mendapatkan versi terbaru dari paket sumber asli dari arsip hulu (*upstream*).

Gambar 3.14: Contoh berkas rules

3.3.3. Berkas-berkas Tambahan dalam direktori "debian"

Di dalam direktori **debian**, selain terdapat berkas-berkas wajib (berkas **control**, **copyright**, **changelog** dan **rules**) juga terdapat berkas-berkas lain yang sifatnya opsional yang dapat membantu pembangunan paket. Berkas-berkas tersebut adalah:

1) **README.Debian**, berkas ini berisi perincian tambahan atau perbedaan antara paket asli atau paket hulu dengan versi paket yang bersangkutan.
2) **compat**, berkas ini berisi nilai tingkat kompatibilitas dari perkakas **debhelper** yang ditetapkan.
3) **conffiles**, berkas ini berisi daftar berkas konfigurasi yang digunakan oleh paket. Biasanya berkas ini akan dibuat jika ada berkas konfigurasi yang disimpan diluar direktori **/etc** (hal ini dikarenakan perkakas **dh_installdeb** hanya akan menandai secara otomatis suatu berkas sebagai **conffiles** yang berada di dalam direktori **/etc**).
4) **nama_paket.cron.***, berkas-berkas ini digunakan untuk mengatur penjadwalan tugas-tugas rutin yang dibutuhkan oleh paket pada waktu-waktu tertentu (seperti setiap satu jam sekali, harian, mingguan, bulanan, atau pada waktu lainnya yang diinginkan). Berkas-berkas ini biasanya berupa berkas *shell script*, kecuali berkas **nama_paket.cron.d** yang mengikuti format **crontab**. Format nama berkas yang digunakan adalah:
   a) **nama_paket.cron.hourly**, nama berkas ini digunakan apabila tugas dijalankan setiap satu jam sekali. Setelah paket terpasang berkas ini akan disimpan sebagai berkas **/etc/cron.hourly/nama_paket**.
   b) **nama_paket.cron.daily**, nama berkas ini digunakan apabila tugas dijalankan setiap satu hari sekali. Setelah paket terpasang berkas ini akan disimpan sebagai berkas **/etc/cron.daily/nama_paket**.
   c) **nama_paket.cron.weekly**, nama berkas ini digunakan apabila tugas dijalankan setiap satu minggu sekali. Setelah paket terpasang berkas ini akan disimpan sebagai berkas **/etc/cron.weekly/nama_paket**.
   d) **nama_paket.cron.monthly**, nama berkas ini digunakan apabila tugas dijalankan setiap satu bulan sekali. Setelah paket terpasang berkas akan disimpan sebagai berkas **/etc/cron.monthly/nama_paket**.
   e) **nama_paket.cron.d** nama berkas ini digunakan apabila tugas dijalankan pada waktu lain yang diinginkan. Setelah paket terpasang berkas ini akan disimpan sebagai berkas **/etc/cron.d/nama_paket**.
5) **dirs**, berkas ini berisi daftar nama direktori yang dibutuhkan tetapi direktori tersebut tidak dibuat oleh prosedur pemasangan normal. Hal ini biasanya disebabkan karena ada madalah dengan **Makefile**.
6) **nama_paket.doc-base**, berkas ini berisi daftar berkas dokumentasi yang dimiliki oleh paket selain yang berupa halaman manual dan info. Berkas dokumentasi ini biasanya berupa berkas **HTML**, **PostScript** (**PS**), atau **PDF**. Berkas-berkas ini nantinya akan dipasang dalam direktori **/usr/share/doc/nama_paket**.
7) **docs**, berkas ini berisi daftar nama berkas dokumentasi yang berasal dari berkas arsip kode sumber hulu. Berkas-berkas ini nantinya akan dipasang dalam direktori **/usr/share/doc/nama_paket** oleh perkakas **dh_installdocs**.
8) **nama_paket.examples**, berkas ini berisi daftar nama berkas *example a*tau berkas contoh.
9) **nama_paket.init**, berkas ini berisi *script* yang digunakan untuk memulai dan menghentikan program **daemon** yang dibawa oleh paket. Berkas ini nantinya akan dipasang sebagai berkas *script* **/etc/init.d/nama_paket**.
10) **nama_paket.default**, berkas ini nantinya akan dipasang sebagai berkas **/etc/default/nama_paket**. Bekas ini menetapkan nilai baku yang digunakan oleh paket. Berkas ini dapat berupa nilai baku konfigurasi yang digunakan paket atau nilai baku yang digunakan oleh suatu *script* **init** (terutama jika *sc-*

ript **init** memiliki fitur-fitur tertentu yang dapat dikonfigurasi atau konfigura-bel).

11) **install**, berkas ini bersisi daftar nama berkas dan nama direktori tujuan ber-kas yang digunakan untuk memasang suatu berkas ke dalam paket yang ti-dak dapat dilakukan menggunakan perintah **make install** standar. Berkas-berkas tersebut akan dipasang menggunakan perkakas **dh_install**.
12) **nama_paket.info**, berkas ini berisi daftar nama berkas info yang terdapat di dalam paket yang digunakan oleh perkakas **dh_installinfo** untuk memasang berkas-berkas info tersebut.
13) **nama_paket.links**, berkas ini berisi daftar *path* penuh berkas sumber dan tujuan tautan simbolis (*symlink*) tambahan yang dibutuhkan paket.
14) **nama_berkas.lintian-overrides** (untuk paket biner) dan **source/lintian-overrides** (untuk paket sumber), berkas ini berfungsi untuk menenangkan laporan diagnosis yang salah dari perkakas **lintian** dalam kasus **Kebijakan Debian** (**Debian Policy**) memperkenankan pengecualian untuk beberapa aturan. Berkas **nama_berkas.lintian-overrides** ini akan dipasang sebagai berkas **/usr/share/lintian/overrides/nama_paket** menggunakan perkakas **dh_lintian**.
15) **manpage.***, berkas-berkas ini merupakan *template* atau pola dasar halaman manual (*manual pages*) yang dihasilkan oleh perkakas **dh_make**. Berkas-berkas pola dasar manual ini terdiri atas beberapa jenis, yaitu:
    a) **manpage.1.ex**, berkas ini merupakan pola dasar halaman manual yang ditulis menggunakan format **nroff**.
    b) **manpage.sgml.ex**, berkas ini merupakan pola dasar halaman manual yang ditulis menggunakan format **SGML**.
    c) **manpage.xml.ex**, berkas ini merupakan pola dasar halaman manual yang ditulis menggunakan format **XML**.
16) **nama_paket.manpages**, berkas ini berisi daftar halaman manual (*manual pages*) yang dimiliki oleh paket. Berkas-berkas tersebut akan dipasang menggunakan perkakas **dh_installman**.
17) **menu**, berkas ini merupakan berkas menu yang digunakan untuk meluncur-kan program yang dibawa oleh paket.
18) **NEWS**, berkas ini merupakan bentuk lain dari berkas catatan perubahan tambahan yang menjelaskan perubahan yang terdapat di dalam paket. Ber-kas ini akan dipasang menggunakan perkakas **dh_installchangelogs**.
19) berkas-berkas *script* pemelihara paket (*package maintainer scripts*), yaitu berkas-berkas *script* yang diletakkan dalam area *control* paket yang akan di-jalankan oleh sistem manajemen paket ketika pemasangan, peningkatan atau penghapusan paket. Berkas-berkas *script* ini adalah berkas **preinst**, **postinst**, **prerm** dan berkas **postrm**.
20) **nama_paket.symbols**, berkas ini merupakan berkas informasi tambahan tentang pustaka bersama (*shared library*) yang terdapat dalam paket.
21) **watch**, berkas ini berisi alamat *URL* situs web tempat mendapatkan berkas asli kode sumber hulu (*upstream*) perangkat lunak, yang digunakan untuk memantau rilis terbaru dari perangkat lunak tersebut.
22) **source/format**, berkas ini berisi sebuah baris kata yang menunjukkan for-mat yang diinginkan untuk paket sumber.
23) **source/options**, berkas ini berisi opsi dari perintah atau perkakas **dpkg-source**. Tujuannya adalah untuk memudahkan penggunan atau pemelihara paket dengan menghilangkan keperluan untuk mengetik opsi perintah **dpkg-**

source tersebut saat pembangunan atau pembangunan kembali paket sumber.

24) **patches/***, berkas-berkas ini merupakan kumpulan berkas tambalan (*patch*) untuk diterapkan pada kode sumber hulu (*upstream*) perangkat lunak. Berkas-berkas tambalan (*patch*) ini dikelola oleh perkakas **quilt**.

3.3.4. Perbandingan di antara Format Paket Sumber Debian

Seiring perkembangan zaman, hampir segala hal yang ada di dunia ini bergerak, berubah dan berkembang, terutama dalam dunia teknologi informasi dan komputer. Sebagaimana paket biner **Debian**, paket sumber **Debian** ini juga turut berkembang. Hal ini ditandai dengan lahirnya beberapa versi format paket sumber **Debian** yang dapat kita gunakan sampai dengan saat ini dan mungkin perlu untuk kita ketahui. Format-format paket sumber **Debian** tersebut adalah format **1.0**, **2.0**, **3.0** (**native**), **3.0** (**quilt**), **3.0** (**custom**), **3.0** (**git**) dan **3.0** (**bzr**).

3.3.4.1. Format: 1.0

Format paket sumber **1.0** ini terdiri atas dua tipe, yaitu:

1) paket *native*, yaitu paket yang berisi perangkat lunak yang secara khusus dikembangkan melalui **Proyek Debian**, seperti **dpkg**, **APT** dan lain sebagainya. Dalam tipe paket ini, paket sumber terdiri atas dua berkas, yaitu berkas **Debian Source Control** (**.dsc**) dan berkas arsip kode sumber perangkat lunak biasanya menggunakan ekstensi nama berkas **.tar.gz** (tanpa ada tambahan nama **.orig**).
2) paket *non-native*, yaitu paket perangkat lunak yang berasal dari luar **Proyek Debian**. Dalam tipe paket ini, paket sumber terdiri atas tiga berkas, yaitu:
   a) berkas **Debian Source Control** (**.dsc**).
   b) berkas **.orig.tar.gz**, berkas ini merupakan sebuah berkas arsip yang berisi kode sumber yang disediakan oleh pengembang aslinya (pengembang hulu atau *upstream*).
   c) berkas **.diff.gz**, berkas ini berisi berkas tambalan (*patch*) yang digunakan untuk menambahkan perubahan-perubahan spesifik **Debian**. Tambalan (*patch*) ini memiliki dua fungsi yaitu untuk membuat berkas-berkas yang dibutuhkan dalam sub-direktori “**debian**” dan untuk menerapkan perubahan pada kode sumber hulu (*upstream*). Dalam format paket sumber yang baru seperti dalam format **3.0** (**quilt**), berkas tambalan (*patch*) **.diff.gz** ini telah digantikan dengan sebuah berkas arsip **.debian.tar.gz** yang berisi perintah-perintah kompilasi dan sekumpulan tambalan (*patch*) untuk kode sumber hulu (*upstream*) yang diberikan oleh pemelihara paket.

Format ini meskipun sebagai format paket sumber lama, namun sampai saat buku ini ditulis masih tetap digunakan sebagai format standar yang digunakan di repositori resmi **Ubuntu Linux**.

3.3.4.2. Format: 2.0

Format **2.0** ini merupakan spesifikasi pertama dari format paket sumber generasi baru yang juga dikenal dengan sebutan “**wig&pen**” (rambut palsu dan pena). Format ini tidak dianjurkan untuk digunakan secara luas dan telah digantikan oleh format **3.0** (**quilt**). Ektraksi paket sumber format ini telah didukung se-

jak **dpkg 1.13.9**, sedangkan pembangunannya didukung sejak **dpkg 1.14.8**. Perilaku format ini hampir sama dengan format **3.0** (**quilt**), kecuali format ini tidak menggunakan daftar tambalan (*patch*) yang jelas. Semua berkas yang berada dalam direktori **debian/patches** harus dicocokkan dengan *regular expression* **Perl** untuk menjadi tambalan (*patch*) yang sah atau benar yang diterapkan pada saat proses ekstraksi. Saat pembangunan paket sumber yang baru, perubahan terhadap kode sumber hulu (*upstream*) akan disimpan dalam sebuah berkas tambalan (*patch*) dengan nama **zz_debian-diff-auto**.

3.3.4.3. Format: 3.0 (native)

Format **3.0** (**native**) ini merupakan bentuk perluasan dari format paket sumber *native* sebagaimana yang telah dijelaskan dalam dalam format **1.0**. Format ini telah mendukung lebih banyak format kompresi (seperti **gzip**, **bzip2**, **xz** dan **lzma**) dan akan mengesampingkan berkas-berkas dan direktori yang spesifik dari suatu sistem pengaturan versi (*Version Controlling System* atau **VCS**) sebagaimana berkas-berkas sementara (temporer) yang lain. Format ini didukung sejak **dpkg 1.14.7**.

3.3.4.4. Format: 3.0 (quilt)

Format **3.0** (**quilt**) ini merupakan format pengganti dari varian *non-native* dari format **1.0** yang didukung sejak **dpkg 1.14.7**. Paket sumber dalam format ini paling sedikit berisi sebuah berkas arsip tarbal asli (**.orig.tar.gz**) dan sebuah berkas arsip tarball **debian** (**.debian.tar.gz**). Paket sumber juga dapat berisi berkas-berkas arsip tarball asli tambahan (**.orig-”komponen”.tar.gz**). Dibandingkan format sebelumnya, format ini memiliki beberapa fitur tambahan, yaitu:

1) mendukung lebih banyak format kompresi, yaitu **gzip**, **bzip2**, **xz** dan **lzma**.
2) mendukung penggunaan banyak berkas arsip tarball hulu (*upstream*).
3) mendukung pencantuman berkas biner.
4) mengganti direktori “**debian**” terkini di dalam berkas arsip tarball hulu (*upstream*) secara otomatis (tidak diperlukan pemaketan ulang).
5) membuat sebuah berkas tambalan (*patch*) yang dapat dikelola oleh perkakas **quilt** dalam direktori **debian/patches/** ketika ditemukan adanya perubahan terhadap berkas-berkas hulu (*upstream*).

3.3.4.5. Format: 3.0 (custom)

Format **3.0** (**custom**) ini merupakan format khusus. Format ini tidak mewakili sebuah format paket sumber yang nyata, tetapi dapat digunakan untuk membuat paket sumber dengan berkas-berkas yang berubah-ubah. Format ini didukung sejak **dpkg 1.14.17**.

3.3.4.6. Format: 3.0 (git)

Format **3.0** (**git**) ini masih dalam tahap percobaan. Paket sumber dalam format ini berisi sebuah berkas **.git** yang merupakan sebuah bundel dari sebuah repositori **git** untuk memegang sumber paket. Format ini didukung sejak **dpkg 1.14.17**.

3.3.4.7. Format: 3.0 (bzr)

Format **3.0** (**bzr**) ini sebagaimana format **3.0** (**git**) juga masih dalam tahap percobaan. Format ini menghasilkan sebuah berkas arsip tarball tunggal yang berisi repositori **bazaar** (**bzr**). Format ini didukung sejak **dpkg 1.14.17**.

3.4. Hubungan di antara Paket-paket Debian

Hampir setiap paket memiliki hubungan atau keterkaitan dengan paket-paket yang lain. Hubungan-hubungan ini dapat terjadi dikarenakan berbagai macam sebab, seperti karena adanya ketergantungan pada paket yang lain, penyediaan layanan (*service*) tertentu, atau karena ketidakcocokan, konflik, atau dapat menggantikan paket yang lain jika dipasang secara bersama-sama. Oleh karena itu, dalam **Sistem Manajemen Paket Debian** ditentukan aturan untuk menyatakan hubungan dengan paket-paket yang lain tersebut dalam beberapa kolom (*field*) yang terdapat dalam berkas **control** paket, baik di dalam paket biner maupun paket sumber. Berbagai hubungan tersebut adalah hubungan ketergantungan, ketidakcocokan, konflik, menggantikan paket lainnya dan penyediaan suatu layanan tertentu.

3.4.1. Ketergantungan

Hubungan ketergantungan atau yang lebih dikenal dengan "**dependensi**" ini menunjukkan bahwa suatu paket akan dapat bekerja dengan baik jika paket yang lain telah dipasang di dalam sistem, baik dengan mempengaruhi secara langsung (perangkat lunak tidak akan dapat digunakan sama sekali jika paket lain tersebut tidak dipasang), menambah fungsionalitas paket tersebut, maupun dengan melengkapi dan meningkatkan nilai guna dari masing-masing paket. Hubungan ketergantungan ini di dalam berkas **control** paket dinyatakan dalam beberapa kolom (*field*) yang masing-masing kolom (*field*) tersebut memiliki tingkat ketergantungan dan kepentingan yang berbeda-beda. Kolom-kolom (*field*) tersebut adalah:

1) **Depends**, kolom ini berisi daftar paket-paket yang diminta dan harus dipasang agar pemasangan paket dapat berhasil dan paket dapat berfungsi dengan baik.
2) **Recommends**, kolom ini berisi daftar paket-paket yang tidak wajib dipasang walaupun memiliki tingkat kepentingan yang tinggi, karena akan meningkatkan fungsionalitas yang ditawarkan oleh paket walaupun sebenarnya tidak diperlukan dalam operasi.
3) **Suggests**, kolom ini berisi daftar paket-paket yang dapat melengkapi dan meningkatkan kegunaan dari masing-masing paket, namun tidak akan terjadi masalah jika tidak kita pasang.
4) **Enhances**, kolom ini berisi daftar paket-paket yang disarankan, tetapi dalam konteks yang berbeda. Keuntungannya adalah dimungkinkan untuk menambahkan saran tanpa harus memodifikasi paket yang bersangkutan. Jadi, semua *add-on*, *plug-in* dan ekstensi lain dari program nantinya dapat muncul dalam daftar saran yang berkaitan dengan perangkat lunak.
5) **Pre-Depends**, kolom ini berisi daftar paket-paket yang dibutuhkan (dependensi), namun memiliki ketentuan yang lebih ketat. Paket ini harus dipasang secara lengkap dan dikonfigurasi dengan benar sebelum proses pemasangan paket yang menyatakan sebagai "**Pre-Depends**" dimulai (yaitu sebelum dilakukannya eksekusi *script* **preinst** dan konfigurasi).

Dalam setiap hubungan ketergantungan atau dependensi ini, kita dimungkinkan untuk membatasi jangkauan versi paket dependensi tersebut. Sebagai contoh, jika kita membutuhkan paket **libc6** dalam versi yang sama atau lebih besar dari "**2.15**", maka dapat kita tulis dengan "**libc6 (> = 2.3.4)**". Operator perbandingan versi yang digunakan adalah sebagai berikut:

- "**<<**" = lebih kecil;
- "**<=**" = lebih kecil atau sama dengan;
- "**=**" = sama dengan (catatan nilai "**2.6.1**" tidak sama dengan "**2.6.1-1**");
- "**>=**" = lebih besar atau sama dengan; dan
- "**>>**" = lebih besar.

Dalam daftar ketergantunagan atau dependensi yang harus dipenuhi, tanda koma ("**,**") berfungsi sebagai pemisah yang berarti "**dan**". Sedangkan tanda garis tegak ("**|**") memiliki arti "**atau**". Dengan demikian, jika terdapat ketergantungan pada "(A atau B) dan C" maka dapat kita tulis dengan "**A | B, C**". Sebaliknya, ekspresi "A atau (B dan C)" harus ditulis sebagai "(A atau B) dan (A atau C)", karena kolom (*field*) **Depends** tidak menoleransi tanda kurung yang mengubah urutan prioritas antara operator logika "**atau**" dan "**dan**", maka dalam kasus tersebut akan ditulis dengan **A | B, A | C**.

Sistem ketergantungan atau dependensi ini merupakan sebuah mekanisme yang baik untuk menjamin operasi atau kerja dari sebuah program perangkat lunak. Selain itu, sistem ketergantungan ini juga memiliki kegunaan lain jika digunakan bersama dengan "***meta-package***" atau paket meta. Paket meta ini adalah sebuah paket kosong yang hanya menggambarkan tentang paket-paket ketergantungan atau dependensi. Paket meta ini memfasilitasi pemasangan kelompok program tertentu. Contoh jika kita menjalankan perintah "**apt-get install nama_paket_meta**", maka secara otomatis akan memasang semua program yang menjadi ketergantungan atau dependensi dari paket meta tersebut. Paket **gnome**, **kde-full** dan **linux-image-amd64** merupakan beberapa contoh dari paket meta.

3.4.2. Konflik atau Pertentangan

Hubungan konflik ini menunjukkan bahwa suatu paket tidak dapat dipasang bersama dengan paket yang lain. Hal ini dapat dikarenakan kedua paket berisi berkas dengan nama yang sama, paket menyediakan layanan (*service*) yang sama pada *port* **TCP** yang sama, atau jika paket dipasang akan menghambat operasi atau kerja dari masing-masing program. Hubungan konflik ini akan dinyatakan dalam kolom (*field*) **Conflicts** di dalam berkas **control** paket **Debian**.

perkakas manajemen paket akan menolak untuk memasang sebuah paket jika paket tersebut dapat memicu konflik dengan paket yang telah terpasang. Namun, terkecuali jika paket yang baru menentukan bahwa paket akan menggantikan paket telah terpasang tersebut, maka perkakas manajemen paket akan memilih untuk menggantikan paket yang lama dengan paket yang baru. Perkakas manajemen paket akan selalu mengikuti instruksi atau perintah yang kita berikan. Jika kita lebih memilih untuk memasang paket yang baru, maka secara otomatis perkakas manajemen paket akan menawarkan untuk menghapus paket yang dapat menimbulkan masalah.

3.4.3. Ketidakcocokan

Hubungan ketidakcocokan ini memiliki efek yang serupa dengan konflik, namun memiliki makna yang khusus. Hubungan ini menandakan bahwa pemasangan suatu paket akan merusak paket yang lain (atau versi tertentu dari paket tersebut) atau secara khusus hubungan ini mengacu pada ketidakcocokan versi. Hubungan ini akan dinyatakan dalam kolom (*field*) **Breaks** dalam berkas **control** paket **Debian**. Perkakas manajemen paket akan menolak untuk memasang paket yang merusak paket yang telah terpasang dan akan mencoba untuk mengatasi masalah dengan memperbaharui paket yang akan menjadi rusak ke versi yang lebih baru (yang diasumsikan masalah tersebut telah teratasi sehingga paket kembali kompatibel). Hal ini dapat terjadi dalam kasus pembaruan suatu program yang tidak memiliki kompatibilitas mundur (*backward compatibility*) seperti jika di versi program yang baru tidak lagi mendukung sejumlah fungsi yang ada dalam versi program sebelumnya dan dapat menyebabkan kerusakan atau gangguan fungsi pada program lain dengan tanpa membuat adanya ketentuan khusus. Kolom (*field*) **Breaks** ini akan mencegah pengguna dari masalah tersebut.

3.4.4. Menggantikan Paket

Hubungan menggantikan ini menunjukkan bahwa paket jika dipasang akan menimpa berkas-berkas yang terdapat dalam paket yang lain yang telah terpasang atau dapat menggantikan seutuhnya paket yang lain yang telah terpasang. Hal ini dikarenakan suatu paket berisi sejumlah berkas yang sama dengan yang terdapat dalam paket yang lain, sehingga jika paket tersebut dipasang maka akan berkas dari paket tersebut akan menimpa berkas yang disediakan paket yang lain. Atau bisa juga dikarenakan adanya perubahan nama paket, isi dari suatu paket juga termasuk dalam paket yang lain, atau adanya pemecahan sebuah paket menjadi beberapa paket dalam versi paket yang baru. Hubungan ini dinyatakan dalam kolom (*field*) **Replaces** dalam berkas **control** paket **Debian**.

3.4.5. Penyediaan Sesuatu

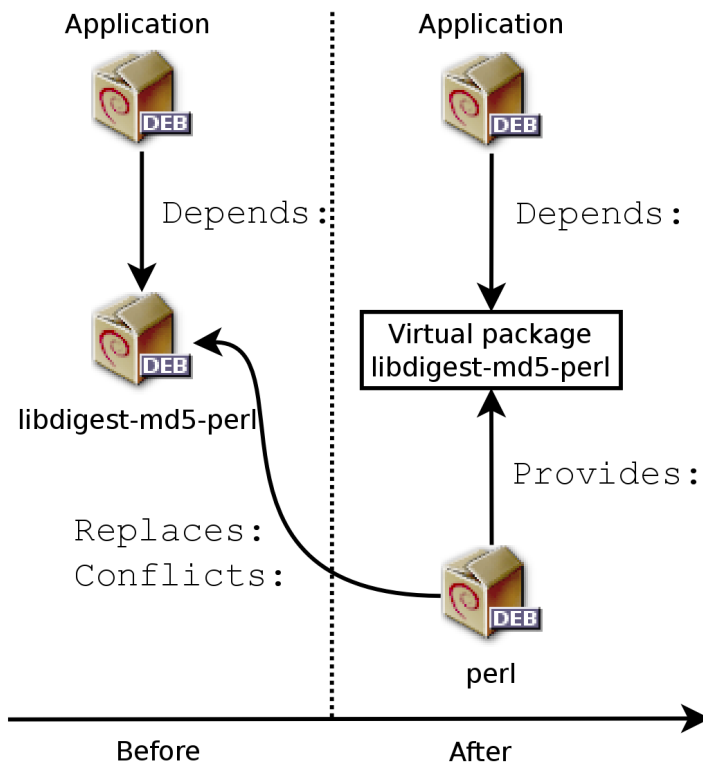
Hubungan menyediakan sesuatu ini menunjukkan bahwa paket menyediakan sesuatu hal yang dikenal sebagai paket virtual (*virtual package*). Paket virtual ini sendiri adalah nama generik yang berlaku untuk salah satu dari sekelompok paket, yang semuanya menyediakan fungsionalitas dasar yang sama. Paket virtual ini secara fisik tidak pernah ada, mereka hanya dimaksudkan untuk mengenali atau mengidentifikasi paket-paket nyata berdasarkan pada kriteria umum dan logis (layanan yang disediakan, kompatibilitas dengan suatu standar program, atau paket yang belum tersedia dan lain sebagainya). Sebagai contoh, program **sendmail**, **postfix** dan **smail** adalah termasuk dalam kelompok program pengirim surat elektronik atau *email* (**mail-transport-agent**). Jika kita memasang salah satu dari kedua program tersebut maka akan dapat memenuhi ketergantungan atau dependensi dari program yang membutuhkan dependensi berupa **mail-transport-agent** di dalam sistem. Sehingga dapat dikatakan bahwa kedua program tersebut menyediakan paket virtual yang bernama **mail-transport-agent**. **Debian** telah menyediakan sebuah mekanisme yang mengatur jika terdapat lebih dari satu paket yang menyediakan paket virtual yang sama dipasang pada sistem. Dengan adanya mekanisme ini, seorang administrator sistem

dapat menetapkan salah satu paket sebagai paket pilihan utama. Mekanisme tersebut adalah dengan menggunakan perintah **update-alternatives**.

Nama paket virtual ini tidak boleh kita buat secara sembarangan karena pengembang **Debian** telah membuat standar nama paket virtual ini yang termuat dalam sebuah **Kebijakan Debian** (**Debian Policy**). Sebagai contoh paket virtual **www-browser** digunakan untuk program peramban web, **x-terminal-emulator** untuk program *emulator* terminal dalam mode grafis dan lain sebagainya. Daftar lengkap nama paket virtual ini dapat kita baca dalam pranala http://www.debian.org/doc/packaging-manuals/virtual-package-names-list.txt .

Selain itu, paket virtual ini juga dapat digunakan untuk menggantikan paket lain yang pada saat ini sudah tidak tersedia dikarenakan fungsi atau isi dari paket tersebut digabungkan ke dalam paket lain. Sebagai contoh, modul **libdigest-md5-perl** merupakan modul tambahan milik **Perl** versi **5.6** dan sekarang telah digabungkan sebagai standar dalam **Perl** versi **5.8** atau yang lebih baru. Oleh karena itu, di dalam paket **Perl** sejak versi **5.8** dicantumkan kolom (*field*) "**Provides: libdigest-md5-perl**". Sehingga jika ada paket yang bergantung pada **libdigest-md5-perl** ini, akan tetap terpenuhi dependensi atau ketergantungannya jika di sistem telah terpasang paket **Perl** versi 5.8 atau yang lebih baru.

Fitur ini jelas sangat berguna, karena kita dapat lebih mudah menyesuaikan diri terhadap proses pengembangan perangkat lunak yang sangat dinamis, seperti perubahan nama paket atau adanya perangkat lunak usang (yang tidak dikembangkan lagi).

Selain kegunaan yang telah disebutkan tadi, paket virtual ini juga memiliki sejumlah keterbatasan. Keterbatasan yang paling utama adalah tidak adanya nomor versi. Contoh jika ada suatu paket yang memiliki ketergantungan terhadap **libdigest-md5-perl** (>= 1.6), adanya paket **Perl 5.10** walaupun tidak akan dapat memenuhi dependensi atau ketergantungannya, namun dalam kenyataannya manajer paket akan tetap menganggapnya telah dapat memenuhi kebutuhan ketergantungan atau dependensi tersebut. Hal ini dikarenakan perkakas manajemen paket akan memilih risiko yang paling rendah, walaupun dengan asumsi versi yang tersedia sebenarnya tidak cocok.

Gambar 3.15: Penggunaan kolom (field) Provides untuk mengabaikan ketergantungan
Sumber: debian-handbook.info

3.4.6. Hubungan antara Paket Sumber dan Paket Biner

Hubungan antar paket tidak hanya terdapat pada paket-paket dengan tipe yang sama, namun juga terdapat pada paket-paket yang berbeda tipe, seperti halnya hubungan antara paket sumber dengan paket biner **Debian**. Hubungan ini terjadi karena pada saat pembangunan paket biner, paket sumber membutuhkan paket-paket biner tertentu telah dipasang atau tidak boleh terpasang. Hubungan ini harus dinyatakan dalam beberapa kolom (*field*) dalam berkas **control** paket sumber. Kolom-kolom tersebut adalah:

1) **Build-Depends**, kolom ini berisi daftar nama paket yang dibutuhkan (telah terpasang dan terkonfigurasi) untuk dapat membangun dari paket sumber (baik ketika membangun paket biner *architecture dependent* maupun paket *architecture independent* serta paket sumber).
2) **Build-Depends-Arch**, kolom ini berisi daftar nama paket yang hanya dibutuhkan ketika membangun paket *architecture dependent*.
3) **Build-Depends-Indep**, kolom ini berisi daftar nama paket yang hanya dibutuhkan ketika membangun paket *architecture independent*.
4) **Build-Conflicts**, kolom ini berisi daftar paket yang tidak boleh dipasang ketika paket sedang dibangun, misalnya karena dapat mengganggu sistem pembangunan paket.
5) **Build-Conflicts-Arch**, kolom ini berisi daftar paket yang tidak boleh dipasang hanya ketika membangun paket *architecture dependent*.

6) **Build-Conflicts-Indep**, kolom ini berisi daftar paket yang tidak boleh dipasang hanya ketika membangun paket *architecture independent*.

4.1. Mengenal dpkg

dpkg atau “**Debian Package Manager**” merupakan perangkat lunak yang merupakan dasar dan inti dari **Sistem Manajemen Paket Debian** yang digunakan dalam sistem operasi **Debian** dan sistem operasi lain yang memilih untuk turut serta menggunakan **Sistem Manajemen Paket Debian** (terutama oleh sistem operasi atau distribusi **GNU/Linux** yang merupakan turunan atau yang pengembangannya berdasar pada distribusi **Debian GNU/Linux**). Di antara perkakas-perkakas yang ada dalam **Sistem Manajemen Paket Debian** ini, perkakas **dpkg** ini merupakan perkakas yang paling utama. Perkakas ini digunakan untuk memasang, membangun, menghapus dan mengelola paket-paket **Debian**. Perkakas ini juga satu-satunya perkakas yang dapat bekerja secara langsung terhadap paket (paket **Debian**).

4.1.1. Sejarah dpkg

Sebagaimana sistem operasi **Debian**, perkakas **dpkg** ini juga memiliki catatan sejarah yang sangat panjang. Pada saat buku ini ditulis, perkakas **dpkg** ini telah mencapai versi **1.18.16** yang dirilis pada tanggal 17 Desember 2016. Berikut adalah sedikit ringkasan sejarah perkakas **dpkg** sampai dengan saat ini:

1) Perkakas **dpkg** ini awalnya dibuat oleh **Ian Murdock** pada tahun 1994 sebagai sebuah *shell* (**sh**) *script* dengan beberapa perkakas **C** *helper* yang dapat digunakan untuk memasang, meningkatkan dan menghapus paket. Semua fitur dasar yang telah tersedia, misalnya:
    a) basis data paket;
    b) skrip-skrip pemelihara paket (skrip **preinst**, **postinst**, **prerm** dan **postrm**);
    c) memisahkan perkakas **dpkg-util** untuk menangani format paket; dan
    d) perkakas **install-info** untuk mengelola berkas **dir**.
2) Cikal bakal dari perkakas **dpkg** ini telah diperkenalkan sejak rilis **Debian 0.91** (29 Januari 1994). Pada saat itu, fitur yang dimiliki oleh perkakas **dpkg** ini masih sangat sederhana, namun telah dapat dijalankan secara interaktif yang dikenal dengan mode pemilihan paket (*package selection mode*). Format berkas paket yang digunakan telah menggunakan ekstensi nama berkas **.deb**, yang sebenarnya merupakan berkas arsip **cpio** yang dikompres menggunakan kompresi **gzip**.
3) Pada bulan Oktober 1994, oleh **Matt Welsh**, **Carl Streeter** dan **Ian Murdock** perkakas **dpkg** ini ditulis ulang, yang awalnya berupa *shell script* diganti menggunakan bahasa pemrograman **Perl**. Bersamaan dengan itu, perkakas **dpkg-util** diubah namanya menjadi **dpkg-deb**. Penulisan ulang perkakas **dpkg** ke dalam bahasa pemrograman **Perl** ini selesai sepenuhnya pada tanggal 28 November 1994 bertepatan pada rilis **dpkg 0.93.10**.

4) Pada tahun 1994 ini juga dimulai pengerjaan sebuah *front-end* dari perkakas **dpkg** ini yang diberi nama “**debian**”, yang kemudian diganti namanya menjadi **dselect**. Versi awal dari perkakas **dselect** ini ditulis menggunakan bahasa pemrograman **Perl**, namun dengan cepat ditulis ulang menggunakan bahasa pemrograman **C++**. Cikal bakal dari perkakas **dselect** ini dimulai pada rilis **dpkg 0.93.12**.
5) Pada saat yang sama dengan **dselect**, perkakas **start-stop-daemon** juga ditambahkan. Perkakas ini digunakan untuk mempermudah dalam menjalankan dan menghentikan program-program “*daemon*” milik sistem.
6) Pada rilis **dpkg 0.93.15** dilakukan perubahan format basis data dan perpindahan basis data dari direktori **/var/adm/** ke direktori **/var/lib/**.
7) Pada bulan April 1995, diperkenalkan format paket sumber **1.0**.
8) Pada bulan Mei 1995 yang bertepatan dengan rilis **dpkg 0.93.50**, diperkenalkan perkakas **dpkg** yang telah ditulis ulang menggunakan bahasa pemrograman **C** oleh **Ian Jackson**. Sebelumnya pada rilis **dpkg 0.93.36** telah ditambahkan dukungan untuk paket virtual (*virtual package*).
9) Pada rilis **dpkg 0.93.72 (**3 September 1995) ditambahkan sistem **alternatives** yaitu sebuah sistem yang digunakan untuk melacak berbagai berkas biner atau program yang menyediakan kegunaan yang serupa.
10) Pada rilis **dpkg 0.93.74** (11 September 1995) ditambahkan perkakas **dpkg-split** yang dapat digunakan untuk memecah berkas paket **Debian** menjadi bagian-bagian yang lebih kecil serta menggabungkannya kembali, untuk mendukung penyimpanan berkas-berkas paket **Debian** yang besar dalam media penyimpanan dengan kapasitas kecil seperti “**floppy disk**’.
11) Pada rilis **dpkg 0.93.76** (20 September 1995) diperkenalkan format paket biner **Debian** yang baru (versi 2.0).
12) **dpkg 1.0.0** dirilis (1 Oktober 1995). Rilis ini merupakan rilis stabil pertama dari perkakas manajemen paket **dpkg**.
13) Pada rilis **dpkg 1.0.10** (14 Januari 1996) ditambahkan perkakas **dpkg-scanpackages**. Perkakas ini digunakan untuk membuat daftar isi dari berkas-berkas yang ada di dalam paket.
14) Pada rilis **dpkg 1.0.11** (19 Januari 1996) ditambahkan perkakas **dpkg-divert**. Perkakas ini digunakan untuk mengatur dan memperbaharui daftar pengalihan (*diversion*).
15) Pada rilis **dpkg 1.0.12** (24 Januari 1996) ditambahkan dukungan untuk “**epoch**”. **epoch** ini adalah bilangan bulat (*integer*) tunggal yang disediakan untuk memperkenankan kekeliruan dalam nomor versi dari paket versi lama dan juga untuk menghilangkan atau mengganti skema penomoran versi sebelumnya yang digunakan paket.
16) Pada rilis **dpkg 1.0.14** (30 Januari 1996) ditambahkan perkakas **dpkg-name**. Perkakas ini digunakan untuk mengganti nama paket **Debian**.
17) **dpkg 1.1.0** dirilis (11 Februari 1996).
18) **dpkg 1.2.0** dirilis (16 Mei 1996).
19) **dpkg 1.3.0** dirilis (6 Agustus 1996).
20) Pada rilis **dpkg 1.3.6** (20 Agustus 1996) ditambahkan perkakas **dpkg-shlibdeps**. Perkakas ini digunakan untuk menghasilkan daftar ketergantungan variabel pengganti dari pustaka bersama (*shared library*).
21) **dpkg 1.4.0** dirilis (12 September 1996). Pada rilis ini berkas paket biner dipecah menjadi dua, yaitu **dpkg** dan **dpkg-dev**..
22) **dpkg 1.4.1** dirilis (1 November 1998).

23) Pada rilis **dpkg 1.4.1.5** (13 Juli 1999) ditambahkan perkakas **dpkg-architecture**. Perkakas ini digunakan untuk mengatur dan mendikte arsitektur untuk pembangunan paket.
24) **dpkg 1.6** dirilis (25 November 1999).
25) Pada rilis **dpkg 1.6.6** (7 Januari 2000) ditambahkan dukungan terhadap **debsign**. **debsign** ini adalah sebuah mekanisme untuk memaksa adanya kebijakan atau aturan keamanan di dalam paket.
26) Pada rilis **dpkg 1.6.7** (16 Januari 2000), perkakas **dpkg** dapat dikompilasi di sistem operasi **HP-UX**, **Solaris** dan **IRIX**.
27) **dpkg 1.7.0** dirilis (5 November 2000). Pada rilis ini ditambahkan perkakas baru **dpkg-statoverride**. Perkakas ini digunakan untuk mengesampingkan kepemilikan dan mode dari suatu berkas.
28) **dpkg 1.8.0** dirilis (25 Desember 2000).
29) **dpkg 1.9.0** dirilis (26 April 2001).
30) **dpkg 1.10** dirilis (21 Juni 2002). Pada rilis ini fungsi, *query* (permintaan informasi) dipisahkan ke dalam perkakas baru, yaitu **dpkg-query**. Beberapa fungsi *query* masih tetap dapat dilakukan menggunakan perkakas **dpkg**, karena **dpkg** juga dapat digunakan sebagai *front-end* dari perkakas **dpkg-query**.
31) **dpkg 1.13.0** dirilis (14 Januari 2005).
32) Pada rilis **dpkg 1.13.9** (12 Juni 2005) ditambahkan dukungan ektraksi paket sumber **format:2.0** (**wig&pen**).
33) **dpkg 1.14.0** dirilis (8 Mei 2007).
34) Pada rilis **dpkg 1.14.8** (19 November 2007) ditambahkan dukungan pembangunan paket sumber **format:2.0** (**wig&pen**).
35) Pada rilis **dpkg 1.14.17** (30 Maret 2008) ditambahkan dukungan terhadap kelompok format paket sumber **3.0**.
36) **dpkg 1.15.0** dirilis (2 Maret 2009).
37) Pada rilis **dpkg 1.15.1** (21 Mei 2009) ditambahkan perkakas **dpkg-vendor**. Perkakas ini digunakan untuk melakukan *query* (permintaan informasi) tentang vendor distribusi.
38) Pada rilis **dpkg 1.15.4** (6 September 2009), perkakas **install-info** bawaan dpkg sebelumnya diganti dengan sebuah *"wrapper'* dari perkakas **GNU install-info**.
39) Pada rilis **dpkg 1.15.7** (21 April 2010) ditambahkan perkakas **dpkg-mergechangelogs** (digunakan untuk melakukan tiga cara penggabungan berkas catatan perubahan (*changelog*) **Debian**), **dpkg-buildflags** (digunakan untuk mengambil *flag* kompilasi dan akan digunakan bersama **debian/rules** untuk meluluskan *flag* kompilasi yang tepat untuk digunakan dalam proses pembangunan paket **Debian** dan **maintscript-helper** dalam skrip pengelola paket (*package maintainer script*) untuk melakukan operasi umum yang bekerja di sekitar keterbatasan yang dimiliki oleh perkakas **dpkg** yang telah diketahui).
40) Pada rilis **dpkg 1.15.7.2** (19 Mei 2010) perkakas **maintscript-helper** diganti namanya menjadi **dpkg-maintscript-helper**.
41) **dpkg 1.16.0** dirilis (1 April 2011).
42) Pada rilis **dpkg 1.16.2** (19 Maret 2012) ditambahkan dukungan terhadap banyak arsitektur (**Multi-Arch**). Dukungan tersebut ditandai dengan penambahan opsi perintah **--add-architecture** dan **--remove-architecture**.
43) **dpkg 1.17.0** dirilis (26 Juli 2013). pada rilis ini *wrapper* dari perkakas **install-info** dihapus.
44) **dpkg 1.18.0** dirilis (18 Mei 2015).

45) **dpkg 1.18.16** dirilis (17 Desember 2016).

Dimulai pada **Debian 4.0**, setiap rilis sistem operasi **Debian** akan memulai siklus versi minor perkakas **dpkg** yang baru. Menimbang bahwa versi **dpkg** tersebut bukan versi awalnya yang dikirimkan pada masing-masing rilis, namun sebagai versi pembaharuan berikutnya untuk mengatasi masalah keamanan atau kutu (*bug*) serius yang mungkin telah keluar. Berikut adalah daftar versi perangkat **dpkg** yang digunakan dalam rilis sistem operasi **Debian** sampai dengan saat ini:

Tabel 4.1: Daftar Versi Perkakas dpkg dalam Rilis Sistem Operasi Debian

No.	*Rilis Debian*	*Nama Kode*	*Versi dpkg*
1	0.93R6	-	1.0.0
2	1.1	buzz	1.2.6
3	1.2	rex	1.4.0.5
4	1.3	bo	1.4.0.8
5	2.0	hamm	1.4.0.32.2
6	2.1	slink	1.4.0.35
7	2.2	potato	1.6.15
8	3.0	woody	1.9.21
9	3.1	sarge	1.10.28
10	4.0	etch	1.13.26
11	5.0	lenny	1.14.31
12	6.0	squeeze	1.15.12
13	7	wheezy	1.16.16
14	8	jessie	1.17.25
15	9 (belum dirilis)	stretch	1.18.x
16	10 (belum dirilis)	buster	1.19.x
17	11 (belum dirilis)	bullseye	1.20.x

4.1.2. Perkakas-perkakas di dalam Paket dpkg

dpkg sebenarnya terdiri atas sejumlah perkakas. Dari paket sumber **dpkg** ini dihasilkan beberapa paket biner dan masing-masing paket biner tersebut (kecuali paket-paket yang berisi berkas pustaka) biasanya menyediakan beberapa perkakas. Paket dan perkakas tersebut meliputi:

1) Paket **dpkg**. Paket ini merupakan paket utama dalam **Sistem Manajemen Paket Debian** dan termasuk dalam kategori paket pokok atau essensial dalam sistem operasi **Debian** dan sistem operasi lain yang turut serta menggu-

nakan **Sistem Manajemen Paket Debian** ini. Paket ini menyediakan perkakas **dpkg** dan beberapa perkakas yang lain. Perkakas-perkakas yang disediakan oleh paket ini meliputi:

a) **dpkg**.
b) **dpkg-deb**. Perkakas ini adalah perkakas yang digunakan untuk melakukan manipulasi terhadap berkas paket (biner) **Debian** (**.deb**).
c) **dpkg-query**. Perkakas ini adalah perkakas yang digunakan untuk melakukan *query* (permintaan informasi) terhadap basis data **dpkg**.
d) **dpkg-split**. Perkakas ini adalah perkakas yang digunakan untuk memecah atau menggabungkan berkas paket (biner) **Debian**.
e) **dpkg-divert**. Perkakas ini adalah perkakas yang digunakan untuk mengesampingkan versi paket dari sebuah berkas.
f) **dpkg-trigger**. Perkakas ini adalah perkakas pemicu paket.
g) **dpkg-statoverride**. Perkakas ini adalah perkakas yang digunakan untuk mengesampingkan kepemilikan dan mode dari berkas-berkas .
h) **dpkg-maintscript-helper**. Perkakas ini adalah perkakas yang bekerja di sekitar pengenalan tentang keterbatasan yang dimiliki **dpkg** dalam *script* pemelihara paket.
i) **update-alternatives**. Perkakas ini adalah perkakas yang digunakan untuk memelihara atau mengelola tautan simbolis (*symbolic link*) yang menetapkan perintah baku (*default*).
j) **start-stop-daemon**. Perkakas ini adalah perkakas yang digunakan untuk menjalankan dan menghentikan program-program “*daemon*” sistem.

2) Paket **dpkg-dev**. Paket ini menyediakan perkakas-perkakas pengembangan paket **Debian** yang dibutuhkan untuk membuka, membangun dan mengunggah paket sumber **Debian**. Perkakas-perkakas tersebut meliputi:

a) **dpkg-source**. Perkakas ini adalah perkakas yang digunakan untuk melakukan manipulasi terhadap berkas paket sumber **Debian** (**.dsc**).
b) **dpkg-name**. Perkakas ini adalah perkakas yang digunakan untuk mengganti nama paket (biner) **Debian** menjadi nama paket yang penuh.
c) **dpkg-scanpackages**. Perkakas ini adalah perkakas yang digunakan untuk membuat berkas daftar penunjuk atau indeks paket (biner).
d) **dpkg-scansources**. Perkakas ini adalah perkakas yang digunakan untuk membuat berkas daftar penunjuk atau indeks paket sumber.
e) **dpkg-shlibdeps**. Perkakas ini adalah perkakas yang digunakan untuk menghasilkan ketergantungan variabel pengganti dari pustaka bersama (*shared library*).
f) **dpkg-buildpackage**. Perkakas ini adalah perkakas yang digunakan untuk membangun paket biner atau paket sumber dari suatu paket sumber.
g) **dpkg-buildflags**. Perkakas ini adalah perkakas yang digunakan untuk mengambil *build flag* atau *flag* kompilasi untuk digunakan selama pembangunan paket.
h) **dpkg-architecture**. Perkakas ini adalah perkakas yang digunakan untuk mengatur dan menentukan arsitektur bagi pembangunan paket.
i) **dpkg-checkbuilddeps**. Perkakas ini adalah perkakas yang digunakan untuk memeriksa ketergantungan dan konflik pembangunan.
j) **dpkg-gencontrol**. Perkakas ini adalah perkakas yang digunakan untuk menghasilkan berkas **control Debian**.
k) **dpkg-genchanges**. Perkakas ini adalah perkakas yang digunakan untuk menghasilkan berkas pengawasan pengunggahan **Debian** (berkas **.changes**).

l) **dpkg-gensymbols**. Perkakas ini adalah perkakas yang digunakan untuk menghasilkan berkas **symbols** (informasi ketergantungan pustaka bersama atau *shared library*).
m) **dpkg-distaddfile**. Perkakas ini adalah perkakas yang digunakan untuk menambahkan entri atau catatan untuk berkas **debian/files**, yaitu berkas yang berisi daftar berkas-berkas yang dihasilkan yang merupakan bagian dari pengunggahan yang sedang dipersiapkan.
n) **dpkg-mergechangelogs**. Perkakas ini adalah perkakas yang digunakan untuk menggabungkan berkas-berkas **debian/changelogs**.
o) **dpkg-parsechangelog**. Perkakas ini adalah perkakas yang digunakan untuk memisahkan berkas-berkas catatan perubahan (*changelog*) **Debian**.
p) **dpkg-vendor**. Perkakas ini adalah perkakas yang digunakan untuk melakukan *query* (permintaan informasi) tentang penyedia (*vendor*) distribusi.

3) Paket **dselect**. Paket ini menyediakan perkakas **dselect** yang merupakan *front-end* dari **dpkg**.
4) Paket **libdpkg-dev**. Paket ini menyediakan berkas-berkas *header* dan pustaka statis yang dibutuhkan untuk pengembangan perangkat lunak yang menggunakan **libdpkg**. **libdpkg** ini merupakan pustaka yang sama yang digunakan secara internal oleh **dpkg**.
5) Paket **libdpkg-perl**. Paket ini menyediakan modul-modul bahasa pemrograman **Perl** yang digunakan oleh *script-script* yang terdapat dalam paket **dpkg-dev**.

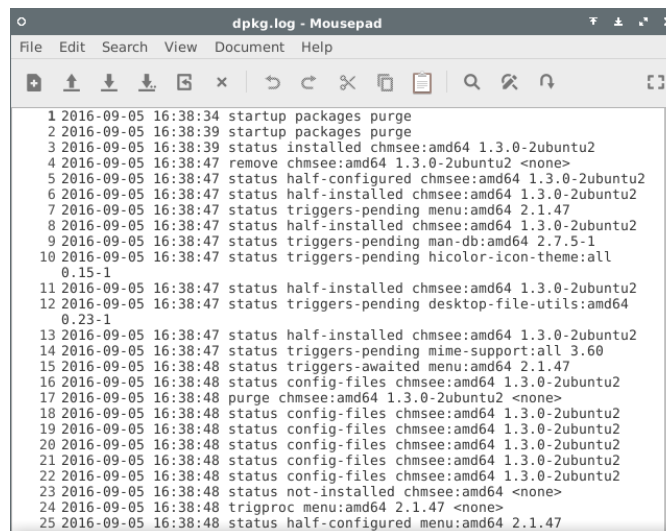
4.1.3. Basis Data dpkg

Basis data (*database*) adalah sekumpulan informasi yang terorganisasi atau teratur, sehingga dapat memudahkan untuk diakses, dikelola dan diperbaharui. Sebagai sebuah perkakas manajer paket, **dpkg** mengumpulkan informasi yang berhubungan dengan paket dan menyimpannya dalam basis data **dpkg**. Basis data **dpkg** terdiri atas beberapa berkas yang merupakan berkas teks sederhana (*plain text*) yang disimpan di dalam direktori **/var/lib/dpkg/** yang merupakan diirektori administratif (**admindir**) baku yang digunakan **dpkg**. Berkas-berkas basis data **dpkg** tersebut meliputi:

1) **/var/lib/dpkg/available**. Berkas ini berisi daftar paket yang tersedia dan dapat dipasang ke dalam sistem.
2) **/var/lib/dpkg/status**. Berkas ini merupakan berkas basis data utama bagi perkakas manajer paket **dpkg**. Berkas ini berisi status dari paket-paket yang tersedia yang berisi informasi tentang paket, seperti apakah paket ditandai untuk dihapus atau tidak, apakah paket telah terpasang atau tidak dan lain sebagainya.
3) **/var/lib/dpkg/statoverride**. Berkas ini berisi daftar terkini dari perencanaan penolakan sistem.
4) **/var/lib/dpkg/diversions**. Berkas ini berisi daftar terkini dari pengalihan sistem.
5) **/var/lib/dpkg/arch**. Berkas ini berisi daftar arsitektur paket yang dapat dipasang oleh **dpkg**.
6) direktori **/var/lib/dpkg/info/**. Direktori ini berisi kumpulan berkas-berkas meta-data (*control information file*) dari paket-paket yang terpasang kecuali berkas “**control**” paket, yaitu berkas-berkas *script* pemelihara paket (*packa-*

ge maintainer scripts), berkas “**conffiles**”, berkas *checksums*, berkas “**list**” serta berkas “**symbols**” dan berkas “**shlibs**”.

4.1.4. Berkas log Perkakas dpkg

Dalam hubungannya dengan dunia komputer, berkas **log** (*logfile*) adalah sebuah berkas yang mencatat setiap kegiatan yang terjadi di dalam sebuah sistem operasi atau perjalanan perangkat lunak lain atau pesan-pesan di antara pengguna yang berbeda dari sebuah perangkat lunak komunikasi. Sebagaimana perangkat lunak yang lain, **dpkg** juga mencatat setiap kegiatan yang dilakukannya dan menyimpannya di dalam sebuah berkas **log**, yaitu di dalam berkas **/var/log/dpkg.log**. Berkas **log** ini sangat ramai, karena merinci setiap tahapan yang dilalui oleh paket saat ditangani oleh **dpkg**. Selain menawarkan cara untuk melacak perilaku dari perkakas **dpkg** ini, berkas **log** juga membantu menjaga riwayat perkembangan sistem (salah satunya adalah dapat menemukan waktu pemasangan atau pembaharuan setiap paket secara tepat) dan informasi ini akan sangat berguna dalam memahami perubahan terbaru dalam perilaku perkakas **dpkg**. Selain itu, semua versi yang telah direkam akan memudahkan dalam pemeriksaan silang (*cross-check*) informasi dengan berkas **changelog.Debian.gz** dari suatu paket, atau bahkan dengan laporan kutu (*bug*) daring.

Gambar 4.1: contoh isi berkas log dpkg

4.2. Informasi tentang Paket

Seperti halnya perkakas manajer paket yang lain, **dpkg** juga memelihara informasi beberapa informasi yang berguna tentang paket-paket yang tersedia. Informasi-informasi ini terbagi atas tiga golongan, yaitu *state* (keadaan), *selection state* (keadaan pemilihan) dan *flag* (bendera). Informasi-informasi ini disimpan **dpkg** dalam basis data yang dapat kita lihat dalam kolom (*field*) **Status** di dalam berkas **/var/lib/dpkg/status** dengan pola sebagai berikut:

```
Status: <selection state> <flag> <state>
```

4.2.1. Package State (Keadaan Paket)

Keadaan paket ini menunjukkan keadaan atau kondisi paket di dalam sistem. Keadaan tersebut terdiri atas beberapa macam keadaan, meliputi:

1) **not-installed**. Keadaan ini menunjukkan bahwa paket tidak terpasang dalam sistem.
2) **config-files**. Keadaan ini menunjukkan bahwa hanya berkas konfigurasi paket yang terdapat di dalam sistem.
3) **half-installed**. Keadaan ini menunjukkan bahwa pemasangan paket telah dimulai, namun tidak sempurna dengan beberapa alasan.
4) **unpacked**. Keadaan ini menunjukkan bahwa paket telah dibuka (*unpack*), namun tidak dikonfigurasi.
5) **half-configured**. Keadaan ini menunjukkan bahwa paket telah dibuka (*unpack*) dan konfigurasi telah dimulai, namun belum sempurna dengan beberapa alasan.
6) **triggers-awaited**. Keadaan ini menunjukkan bahwa paket menunggu pemrosesan pemicu (*trigger*) oleh paket lain.
7) **triggers-pending**. Keadaan ini menunjukkan bahwa paket telah dipicu.
8) **installed**. Keadaan ini menunjukkan bahwa paket telah dibuka (*unpack*) dan dikonfigurasi dengan benar.

4.2.2. Package Selection State (Keadaan Pemilihan Paket)

Keadaan pemilihan paket ini menunjukkan keadaan pemilihan atas suatu paket, seperti suatu paket telah dipilih akan dipasang atau dihapus. Golongan keadaan ini terdiri atas beberapa macam keadaan, meliputi:

1) **install**. Keadaan ini menunjukkan bahwa paket telah dipilih untuk dilakukan pemasangan.
2) **hold**. Keadaan ini menunjukkan bahwa paket telah ditandai dengan tanda ditahan (*hold*) dan tidak sedang ditangani oleh **dpkg**, kecuali jika dalam keadaan terpaksa, maka kita dapat menggunakan opsi –**force-hold**.
3) **deinstall**. Keadaan ini menunjukkan bahwa paket telah dipilih untuk dilakukan penghapusan (menghapus semua berkas paket, kecuali berkas-berkas konfigurasi).
4) **purge**. Keadaan ini ini menunjukkan bahwa paket telah dipilih untuk dibersihkan (menghapus semua hal yang berasal dari paket, termasuk berkas-berkas konfigurasi).

4.2.3. Package Flag (Bendera Paket)

Bendera paket ini terdiri atas dua macam, meliputi:

1) **ok**. Paket yang ditandai dengan tanda **ok** ini menunjukkan bahwa paket dalam keadaan yang telah diketahui, namun mungkin memerlukan pemrosesan lebih lanjut.
2) **reinstreq**. Paket yang ditandai dengan tanda **reinstreq** ini menunjukkan bahwa paket telah rusak dan membutuhkan pemasangan ulang. Paket-paket ini tidak dapat dihapus, kecuali jika terpaksa, maka kita dapat menggunakan opsi –**force-remove-reinstreq**.

Gambar 4.2: Contoh beberapa keadaan paket dalam berkas /var/lib/dpkg/status

4.3. Bekerja dengan dpkg

4.3.1. Parameter Perintah dan Konfigurasi dpkg

dpkg sebagai sebuah perkakas manajer paket dijalankan melalui antarmuka baris perintah (*command line interface*) dengan menggunakan perintah **dpkg**. Perintah **dpkg** ini dikendalikan secara penuh melalui beberapa parameter yang terdiri atas satu buah aksi (*action*) dan dengan tanpa menggunakan opsi (*option*) atau dapat juga dengan tambahan sejumlah opsi. Parameter aksi ini memberitahukan **dpkg** tentang apa yang harus dilakukan, sedangkan parameter opsi mengendalikan perilaku aksi tersebut dalam beberapa cara. **dpkg** juga dapat digunakan sebagai *front-end* dari perkakas **dpkg-deb** dan **dpkg-query**. Pola perintah yang digunakan adalah sebagai berikut:

```
dpkg [<opsi>...] <aksi>
```

Berikut ini adalah daftar dan penjelasan ringkas dari parameter aksi perintah **dpkg** yang dapat kita gunakan:

Tabel 4.2: Daftar Parameter Aksi (Action) Perintah dpkg

No.	**Parameter**	*Format Perintah*	*Kegunaan*
1	`-i, --install`	`dpkg -i, --install <berkas paket>...`	Memasang paket.

No.	**Parameter**	*Format Perintah*	*Kegunaan*
2	`--unpack`	`dpkg --unpack <berkas paket>`	Membuka paket tetapi tidak mengkonfigurasi paket.
3	`--configure`	`dpkg --configure <nama paket>...|-a|--pending`	Mengkonfigurasi paket yang telah dibuka tetapi tidak terkonfigurasi.
4	`--triggers-only`	`dpkg --triggers-only <nama paket>...|-a|--pending`	Hanya memproses pemicu (*trigger*).
5	`-r, --remove`	`dpkg -r, --remove <nama paket>...|-a|--pending`	Menghapus paket yang terpasang. Menghapus semua berkas yang dibawa paket kecuali berkas-berkas konfigurasi.
6	`-P, --purge`	`dpkg -P, --purge <nama paket>...|-a|--pending`	Membersihkan paket yang terpasang atau paket yang telah dihapus. Menghapus semua berkas yang dibawa paket termasuk berkas-berkas konfigurasi.
7	`-V, --verify`	`dpkg -V, --verify <nama paket>...`	Memverifikasi keutuhan atau integritas paket.
8	`-C, --audit`	`dpkg -C, --audit <nama paket>...`	Memeriksa stabilitas atau kenormalan dan konsistensi basis data dari suatu paket tertentu.
9	`--update-avail`	`dpkg --update-avail <berkas data paket>`	Memperbaharui basis data paket yang tersedia (**var/lib/dpkg/available**).
10	`--merge-avail`	`dpkg --merge-avail <berkas data paket>`	Menggabungkan informasi baru ke basis data paket yang tersedia (**var/lib/dpkg/available**).
11	`-A, --record-avail`	`dpkg --record-avail <berkas paket>`	Memperbaharui informasi paket yang terdapat di dalam basis data paket yang tersedia (**var/lib/dpkg/available**) dengan informasi yang berasal dari berkas paket.
12	`--forget-old-unavail`	`dpkg --forget-old-unavail`	Menghapus informasi tentang paket yang tidak tersedia atau yang telah usang dari basis data paket yang tersedia (**var/lib/dpkg/available**).
13	`--clear-avail`	`dpkg --clear-avail`	Menghapus semua informasi paket yang terdapat di dalam basis data paket yang tersedia (**var/lib/dpkg/available**).

No.	Parameter	*Format Perintah*	*Kegunaan*
14	`--get-selections`	`dpkg --get-selections [pola nama paket...]`	Mendapatkan daftar pemilihan paket, dan menuliskannya ke **stdout**.
15	`--set-selections`	`dpkg --set-selections < <berkas teks>`	Mengatur pemilihan paket menggunakan berkas yang dibaca dari **stdin**. Berkas teks ini isinya harus dalam format <nama paket> <keadaan paket>.
16	`--clear-selections`	`dpkg --clear-selections`	Mengatur keadaan (*state*) paket yang diminta dari setiap paket selain paket **Essensial** menjadi **deinstall**.
17	`--yet-to-unpack`	`dpkg --yet-to-unpack`	Mencari paket yang telah dipilih untuk pemasangan, tetapi dengan beberapa alasan masih belum terpasang.
18	`--predep-package`	`dpkg --predep-package`	Menampilkan sebuah paket tunggal yang merupakan target dari satu atau lebih pre-dependensi yang relevan dan paket tersebut tidak memiliki pre-dependensi yang tidak terpenuhi
19	`--add-architecture`	`dpkg --add-architecture <nama arsitektur>`	Menambahkan arsitektur ke daftar arsitektur yang paketnya dapat dipasang.
20	`--remove-architecture`	`dpkg --remove-architecture <nama arsitektur>`	Menghapus arsitektur dari daftar arsitektur yang paketnya dapat dipasang.
21	`--print-architecture`	`dpkg --print-architecture`	Menampilkan nama arsitektur yang paketnya dapat dipasang oleh **dpkg** (arsitektur sistem asli).
22	`--print-foreign-architectures`	`dpkg --print-foreign-architectures`	Menampilkan daftar arsitektur tambahan yang paketnya telah diijinkan untuk dipasang oleh **dpkg**.
23	`--assert-support-predepends`	`dpkg --assert-support-predepends`	Menyatakan bahwa **dpkg** mendukung kolom (*field*) **Pre-Depends**.
24	`--assert-working-epoch`	`dpkg --assert-working-epoch`	Menyatakan bahwa **dpkg** mendukung adanya **epoch** dalam penulisan nomor versi.
25	`--assert-long-filenames`	`dpkg --assert-long-filenames`	Menyatakan bahwa **dpkg** mendukung nama berkas yang panjang.

No.	**Parameter**	*Format Perintah*	*Kegunaan*
26	`--assert-multi-conrep`	`dpkg --assert-multi-conrep`	Menyatakan bahwa **dpkg** mendukung banyak kolom **Conflicts** dan **Replaces**.
27	`--assert-multi-arch`	`dpkg --assert-multi-arch`	Menyatakan bahwa **dpkg** mendukung kolom-kolom (*field*) dan semantik **multi-arch**.
28	`--assert-versioned-provides`	`dpkg --assert-versioned-provides`	Menyatakan bahwa **dpkg** mendukung penomoran versi dalam kolom (*field*) dan **Provides**.
29	`--compare-versions`	`dpkg --compare-versions <versi1> <versi2>`	Membandingkan nomor versi.
30	`-?, --help`	`dpkg -?, --help`	Menampilkan pesan bantuan singkat.
31	`--force-help`	`dpkg ---force-help`	Menampilkan pesan bantuan tentang opsi --force-xxxx.
32	`-Dh, --debug=help`	`dpkg --debug=help`	Menampilkan pesan bantuan tentang opsi awakutu (*debugging*).
33	`--version`	`dpkg --version`	Menampilkan informasi versi **dpkg**.
Aksi yang berkaitan dengan fungsi sebagai *front-end* **dpkg-deb**.			
34	`-b, --build`	`dpkg -b, --build <direktori> [<bekas paket>|<direktori>]`	Membangun paket biner **Debian**.
35	`-c, --contents`	`dpkg -c, --contents <berkas paket>`	Menampilkan daftar isi paket biner **Debian**.
36	`-e, --control`	`dpkg -e, --control <berkas paket> [<direktori>]`	Mengekstrak berkas-berkas **control** atau meta-data dari paket biner **Debian**.
37	`-x, --extract`	`dpkg -x, --extract <berkas paket> <direktori>`	Mengekstrak berkas-berkas yang terkandung dalam paket biner **Debian**.
38	`-X, --vextract`	`dpkg -X, --vextract <berkas paket> <direktori>`	Mengekstrak dan menampilkan berkas-berkas yang dikandung oleh paket biner **Debian**.
39	`-f, --field`	`dpkg -f, --field <berkas paket> [<nama kolom control>...]`	Menampilkan kolom (*field*) **control** dari sebuah paket.
40	`--ctrl-tarfile`	`dpkg --ctrl-tarfile <berkas paket>`	Menampilkan keluaran berkas arsip (**tar**) **control** yang terkandung dalam paket.
50	`--fsys-tarfile`	`dpkg --fsys-tarfile <berkas paket>`	Menampilkan keluaran berkas arsip (**tar**) sistem berkas yang ter-

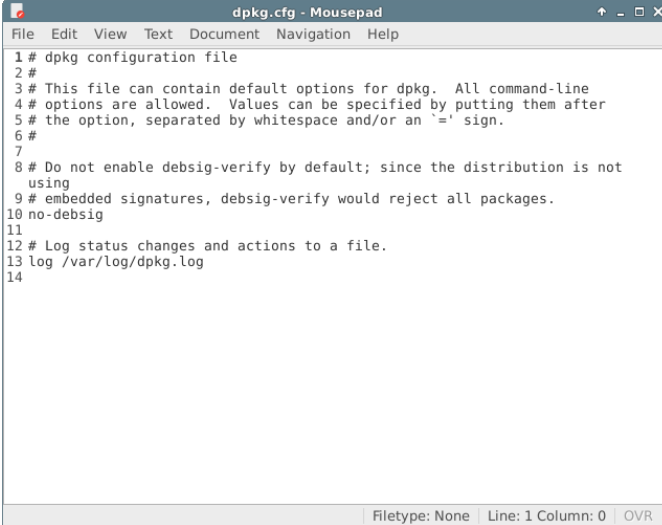
No.	**Parameter**	*Format Perintah*	*Kegunaan*
			kandung dalam paket.
51	`-I, --info`	`dpkg -I, --info <berkas paket> [<nama berkas control>...]`	Menampilkan informasi tentang sebuah paket.
Aksi yang berkaitan dengan fungsi sebagai *front-end* **dpkg-query**.			
52	`-l, --list`	`dpkg -l, --list <pola nama paket>...`	Menampilkan daftar paket yang sesuai dengan pola yang diberikan.
53	`-s, --status`	`dpkg -s, --status <nama paket>...`	Melaporkan status paket yang spesifik.
54	`-L, --listfies`	`dpkg -L, --listfies <nama paket>...`	Menampilkan daftar berkas yang terpasang dalam sistem dari suatu nama paket.
55	`-S, --search`	`dpkg -S, --search <nama paket>...`	Mencari sebuah nama paket dari paket-paket yang terpasang.
56	`-p, --print-avail`	`dpkg -p, --print-avail <nama paket>...`	Menampilkan perincian tentang suatu nama paket sebagaimana yang ditemukan dalam **/var/lib/dpkg/available**.

Sedangkan untuk daftar dan penjelasan ringkas dari parameter opsi perintah **dpkg** yang dapat kita gunakan adalah sebagai berikut:

Tabel 4.3: Daftar Parameter Opsi (Option) Perintah dpkg

No.	**Parameter**	*Kegunaan*
1	`--abort-after=<angka>`	Mengganti ketentuan setelah seberapa banyak kekeliruan (*error*) yang akan menyebabkan **dpkg** dibatalkan. Nilai bakunya adalah "**50**".
2	`-B, --auto-deconfigure`	Menghapus secara otomatis konfigurasi dari paket yang bergantung pada paket yang telah dihapus.
3	`-D<bil. oktal>, --debug=<bil. oktal>`	Menghidupkan awakutu (*debugging*). Penjelasan nilai bilangan oktal adalah sebagai berikut: 1 = secara umum informasi kemajuan (progres) yang sangat membantu 2 = pemanggilan dan status dari skrip pengelola 10 = keluaran untuk setiap berkas yang telah diproses 100 = banyak keluaran untuk setiap berkas yang telah diproses 20 = keluaran untuk setiap berkas konfigurasi 200 = banyak keluaran untuk setiap berkas konfigurasi 40 = ketergantungan dan konflik 400 = banyak keluaran ketergantungan/konflik 10000 = aktivasi dan pemrosesan pemicu (*trigger*)

No.	Parameter	*Kegunaan*
		20000 = banyak keluaran mengenai pemicu 40000 = jumlah yang tak masuk akal dari keluaran mengenai pemicu 1000 = banyak ruang kosong (*drivel*), sebagai contoh tentang **dpkg/info dir** 2000 = jumlah yang tak masuk akal dari ruang kosong (*drivel*)
4	`--force-all`	Menghidupkan semua opsi pemaksaan.
5	`--force-downgrade`	Memaksa untuk memasang paket, bahkan jika versi baru paket tersebut telah terpasang.
6	`--force-configure-any`	Memaksa untuk juga mengkonfigurasi suatu paket yang telah dibuka namun tidak terkonfigurasi yang menjadi ketergantungan dari suatu paket yang saat ini akan dipasang.
7	`--force-hold`	Memaksa untuk memproses paket bahkan jika paket ditandai untuk ditahan (*hold*).
8	`--force-remove-reinstreq`	Memaksa untuk menghapus paket bahkan jika paket rusak dan ditandai “butuh pemasangan ulang”.
9	`--force-remove-essential`	Memaksa untuk menghapus paket bahkan jika paket dianggap sangat penting (**Essential**).
10	`--force-depends`	Mengubah semua masalah ketergantungan menjadi peringatan.
11	`--force-depends-version`	Memaksa untuk tidak mempedulikan tentang versi paket ketika pemeriksaan ketergantungan.
12	`--force-breaks`	Memaksa untuk memasang paket bahkan jika paket dapat merusak paket yang lain.
13	`--force-conflicts`	Memaksa untuk memasang paket bahkan jika paket konflik dengan paket yang lain.
14	`--force-confmiss`	Memasang berkas konfigurasi jika berkas konfigurasi saat ini telah hilang tanpa meminta masukan dari pengguna.
15	`--force-confnew`	Selalu memasang berkas konfigurasi baru tanpa meminta masukan dari pengguna. Berkas konfigurasi saat ini akan disimpan dalam berkas dengan akhiran **.dpkg-old**.
16	`--force-confold`	Selalu mempertahankan berkas konfigurasi lama tanpa meminta masukan dari pengguna. Berkas konfigurasi versi baru akan disimpan dalam berkas dengan akhiran **.dpkg-dist**.
17	`--force-confdef`	Selalu memilih tindakan baku tanpa meminta masukan dari pengguna.
18	`--force-confask`	Selalu menawarkan untuk mengganti berkas konfigu-

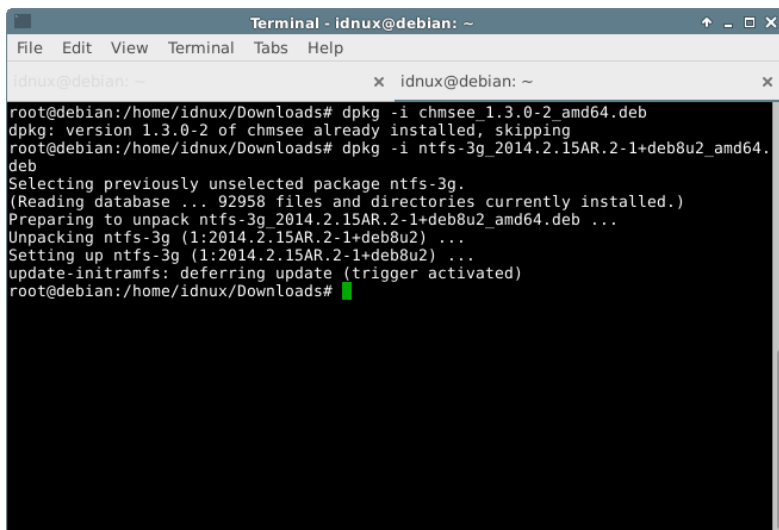
No.	**Parameter**	*Kegunaan*
		rasi dengan versi yang terdapat dalam paket yang sedang dipasang, bahkan jika versi berkas konfigurasi di dalam paket tidak berubah.
19	`--force-overwrite`	Memaksa untuk menimpa berkas paket yang satu dengan berkas paket yang lain.
20	`--force-overwrite-dir`	Memaksa untuk menimpa direktori paket yang satu dengan berkas paket yang lain.
21	`--force-overwrite-diverted`	Memaksa untuk menimpa berkas yang dialihkan dengan versi yang tidak dialihkan.
22	`--force-unsafe-io`	Memaksa untuk tidak melakukan operasi masukan/keluaran (*input/output*) yang aman ketika melakukan pembukaan paket.
23	`--force-architcture`	Memaksa untuk memproses paket bahkan paket dengan arsitektur yang salah atau arsitekturnya sama sekali tidak didukung.
24	`--force-bad-version`	Memaksa untuk memproses paket bahkan paket dengan versi yang salah.
25	`--force-bad-path`	**PATH** program penting hilang, sehingga masalah mungkin terjadi.
26	`--force-not-root`	Mencoba untuk memasang atau menghapus sesuatu bahkan ketika tidak sedang sebagai *super user* (**root**).
27	`--force-bad-verify`	Memaksa untuk memasang paket bahkan jika paket gagal dalam pemeriksaan keasliannya.
28	`--no-force-all`, `--refuse-all`	Mematikan semua opsi pemaksaan.
29	`--no-force-downgrade`, `--refuse-downgrade`	Menolak untuk memasang paket, jika versi baru paket tersebut telah terpasang.
30	`--no-force-configure-any`, `--refuse-configure-any`	Menolak untuk juga mengkonfigurasi suatu paket yang telah dibuka namun tidak terkonfigurasi yang menjadi ketergantungan dari suatu paket yang saat ini akan dipasang.
31	`--no-force-hold`, `--refuse-hold`	Menolak untuk memproses paket jika paket ditandai untuk ditahan (*hold*).
32	`--no-force-remove-reinstreq`, `--refuse-remove-reinstreq`	Menolak untuk menghapus paket jika paket rusak dan ditandai “butuh pemasangan ulang”.
33	`--no-force-remove-essential`, `--refuse-`	Menolak untuk menghapus paket jika paket dianggap sangat penting (**Essential**).

No.	**Parameter**	*Kegunaan*
	essential	
34	--no-force-dep-nds, --refuse-depends	Menolak untuk mengubah semua masalah ketergantungan menjadi peringatan.
35	--no-force-depends-versi-on, --refuse-depends-version	Menolak untuk tidak mempedulikan tentang versi paket ketika pemeriksaan ketergantungan.
36	--no-force-bre-aks, --refuse-breaks	Menolak untuk memasang paket jika paket dapat merusak paket yang lain.
37	--no-force-con-flicts, --re-fuse-conflicts	Menolak untuk memasang paket jika paket konflik dengan paket yang lain.
38	--no-force-con-fmiss, --re-fuse-confmiss	Menolak untuk memasang berkas konfigurasi jika berkas konfigurasi saat ini telah hilang tanpa meminta masukan dari pengguna.
39	--no-force-con-fnew, --refuse-confnew	Menolak untuk selalu memasang berkas konfigurasi baru tanpa meminta masukan dari pengguna.
40	--no-force-con-fold, --refuse-confold	Menolak untuk selalu mempertahankan berkas konfigurasi lama tanpa meminta masukan dari pengguna.
41	--no-force-con-fdef, --refuse-confdef	Menolak untuk selalu memilih tindakan baku tanpa meminta masukan dari pengguna.
42	--no-force-con-fask, --refuse-confask	Menolah untuk selalu menawarkan untuk mengganti berkas konfigurasi dengan versi yang terdapat dalam paket yang sedang dipasang, jika versi berkas konfigurasi di dalam paket tidak berubah.
43	--no-force-ove-rwrite, --re-fuse-overwrite	Menolak untuk menimpa berkas paket yang satu dengan berkas paket yang lain.
44	--no-force-ove-rwrite-dir, --refuse-overw-rite-dir	Menolak untuk menimpa direktori paket yang satu dengan berkas paket yang lain.
45	--no-force-ove-rwrite-diverted, --refuse-overwrite-diverted	Menolak untuk menimpa berkas yang dialihkan dengan versi yang tidak dialihkan.
46	--no-force-uns-afe-io, --re-	Menolak untuk tidak melakukan operasi masukan/keluaran (*input/output*) yang aman ketika melakukan pem-

No.	**Parameter**	*Kegunaan*
	`fuse-unsafe-io`	bukaan paket.
47	`--no-force-architecture, --refuse-architecture`	Menolak untuk memproses paket paket dengan arsitektur yang salah atau arsitekturnya sama sekali tidak didukung.
48	`--no-force-bad-version, --refuse-bad-version`	Menolak untuk memproses paket dengan versi yang salah.
49	`--no-force-bad-path, --refuse-bad-path`	Menolak untuk memproses jika **PATH** program penting hilang, sehingga masalah mungkin terjadi.
50	`--no-force-not-root, --refuse-not-root`	Menolak untuk memasang atau menghapus sesuatu ketika tidak sedang sebagai *super user* (**root**).
51	`--no-force-bad-verify, --refuse-bad-verify`	Menolak untuk memasang paket jika paket gagal dalam pemeriksaan keasliannya.
52	`--ignore-depends=<nama paket>`	Mengabaikan pemeriksaan dependensi untuk paket tertentu (Sebenarnya, pemeriksaan tetap dilakukan, tetapi hanya memberikan peringatan tentang konflik, selainnya tidak).
53	`--no-act, --dry-run, --simulate`	Melakukan semua hal yang seharusnya dapat dilakukan, tetapi tidak menulis perubahan apapun.
54	`-R, --recursive`	Secara berulang (*recursive*) menangani semua berkas biasa yang mencocoki pola **.deb** yang ditemukan pada direktori yang ditentukan dan semua sub-direktorinya.
55	`-G`	Tidak memasang paket jika versi yang baru dari paket yang sama telah terpasang.
56	`--admindir=<direktori>`	Mengganti direktori administratif baku yang berisi banyak berkas yang memberikan informasi tentang status paket terpasang atau terhapus dal lain sebagainya. Bakunya adalah **/var/lib/dpkg/**.
57	`--instdir=<direktori>`	Mengganti direktori pemasangan baku yang merujuk pada direktori di mana paket akan dipasang. Bakunya adalah "**/**".
58	`--root=<direktori>`	Mengganti direktori **root**. Mengganti **instdir** ke <direktori> dan **admindir** ke <direktori>/var/lib/dpkg/**.**
59	`-O, --selected-only`	Hanya memproses paket yang dipilih untuk pemasangan.
60	`-E, --skip-same-version`	Tidak memasang paket paket jika versi yang sama dari paket telah terpasang.

No.	**Parameter**	*Kegunaan*
61	--pre-invoke=-"perintah"	Mengatur perintah *invoke hook* untuk dijalankan sebelum **dpkg** menjalankan aksi **unpack**, **configure**, **install, triggers-only**, **remove** dan **purge**.
62	--post-invoke=-"perintah"	Mengatur perintah *invoke hook* untuk dijalankan setelah **dpkg** menjalankan aksi **unpack**, **configure**, **install, triggers-only**, **remove** dan **purge**.
63	--path-exclude-=<*glob-pattern*>	Mengatur *glob-pattern* sebagai tapis *path* dengan mengeluarkan *path* yang mencocoki pola yang ditetapkan selama pemasangan.
64	--path-include-=<*glob-pattern*>	Mengatur *glob-pattern* sebagai tapis *path* dengan memasukkan ulang *path* yang sebelumnya dikeluarkan yang mencocoki pola yang ditetapkan selama pemasangan.
65	--verify-format <nama format>	Mengatur format keluaran bagi aksi –**verify**.
66	--status-fd "**n**"	Mengirim status dan informasi perkembangan (progres) paket yang dapat dibaca oleh mesin (*machine readable*) ke berkas penggambar atau penjelas "**n**". Contoh perintah: dpkg –status-fd 3 -r chmsee 3>nama.txt
67	--status-logger="perintah"	Mengirim status dan informasi perkembangan (progres) paket yang dapat dibaca oleh mesin (*machine readable*) ke masukan standar perintah *shell*.
68	--log=<nama berkas>	Mencatat perubahan status pembaharuan dan aksi ke <nama berkas> sebagai ganti dari berkas **log** baku **/var/log/dpkg.log**.
69	--no-debsign	Tidak mencoba untuk mencocokkan (verifikasi) tanda tangan (*signature*) paket.
70	--no-triggers	Tidak menjalankan pemicu (*trigger*) tetapi aktivasi pemicu masih dicatat.
71	--triggers	Membatalkan opsi –**no-triggers** sebelumnya.

Semua parameter opsi perintah **dpkg** ini dapat ditetapkan baik melalui baris perintah maupun di dalam berkas konfigurasi **dpkg** yaitu berkas **/etc/dpkg/dpkg.cfg** atau berkas-berkas yang terpecah dalam direktori konfigurasi **dpkg** yaitu direktori **/etc/dpkg/dpkg.cfg.d/**. Ketika kita menetapkan suatu parameter opsi di dalam berkas konfigurasi **dpkg**, maka opsi tersebut akan diberlakukan sebagai opsi baku bagi seluruh aksi yang dilakukan **dpkg** (tergantung opsi tersebut apakah dapat diterapkan pada seluruh aksi **dpkg** atau tidak).

```
dpkg.cfg - Mousepad
File  Edit  View  Text  Document  Navigation  Help
 1 # dpkg configuration file
 2 #
 3 # This file can contain default options for dpkg.  All command-line
 4 # options are allowed.  Values can be specified by putting them after
 5 # the option, separated by whitespace and/or an `=' sign.
 6 #
 7
 8 # Do not enable debsig-verify by default; since the distribution is not
   using
 9 # embedded signatures, debsig-verify would reject all packages.
10 no-debsig
11
12 # Log status changes and actions to a file.
13 log /var/log/dpkg.log
14
Filetype: None | Line: 1 Column: 0 | OVR
```

Gambar 4.3: Isi Baku Berkas Konfigurasi dpkg

Sebagai contoh, jika kita menginginkan **dpkg** untuk tidak menjalankan pemicu (*triggers*) dan tidak memasang paket dengan versi yang sama secara baku, maka kita dapat menambahkan opsi **`no-triggers`** dan **`skip-same-version`** dalam berkas konfigurasi **dpkg**.

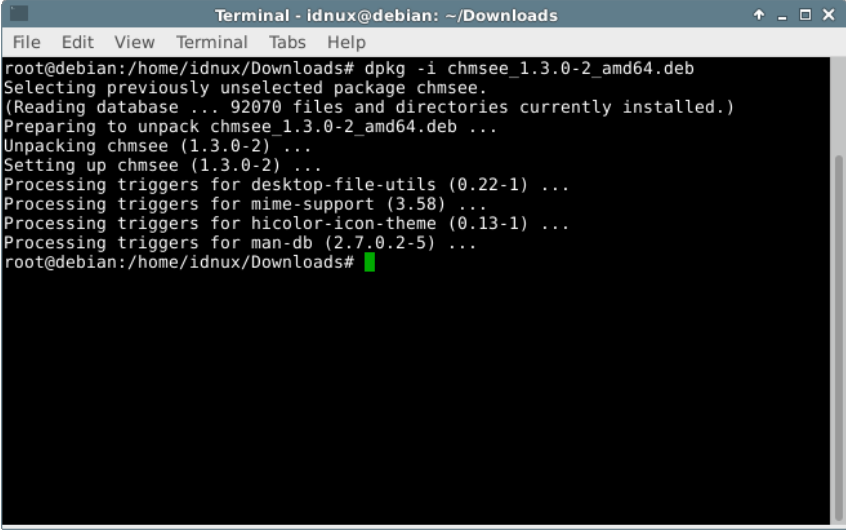
```
dpkg.cfg - Mousepad
File  Edit  View  Text  Document  Navigation  Help
 1 # dpkg configuration file
 2 #
 3 # This file can contain default options for dpkg.  All command-line
 4 # options are allowed.  Values can be specified by putting them after
 5 # the option, separated by whitespace and/or an `=' sign.
 6 #
 7
 8 # Do not enable debsig-verify by default; since the distribution is not
   using
 9 # embedded signatures, debsig-verify would reject all packages.
10 no-debsig
11
12 # Log status changes and actions to a file.
13 log /var/log/dpkg.log
14
15 no-triggers
16 skip-same-version
17
Filetype: None | Line: 1 Column: 0 | OVR
```

Gambar 4.4: Berkas Konfigurasi dpkg yang sudah ditambahkan Opsi Tambahan

Gambar 4.5: Mencoba memasang Paket dengan opsi –no-triggers dan –skip-same-version

Dalam melakukan konfigurasi terhadap perkakas **dpkg** ini, ada beberapa hal yang harus diperhatikan oleh pengguna. Yang pertama adalah tambahkan hanya opsi yang benar-benar sangat kita perlukan. Menggunakan opsi yang baku dalam berkas konfigurasi merupakan pilihan yang paling aman buat sistem. Selain itu, pastikan bahwa kita benar-benar mengetahui kegunaan dan akibat yang mungkin ditimbulkan dari opsi yang akan kita gunakan. Menambahkan opsi-opsi tertentu apalagi jika secara serampangan tanpa mengetahui akibat yang mungkin akan ditimbulkan dapat menyebabkan kerusakan pada sistem. Oleh karena

itu, sangat diperlukan kehati-hatian dari para pengguna sebelum mencoba mengubah isi dari berkas konfigurasi **dpkg** ini.

4.3.2. Memasang Paket

Pemasangan paket merupakan salah satu kegiatan utama dalam kegiatan manajemen paket. Sebagai sebuah perkakas manajer paket, **dpkg** dapat kita gunakan untuk memasang paket yang telah tersedia atau paket lokal ke dalam sistem (**dpkg** tidak memiliki kemampuan untuk mengambil atau mengunduh berkas paket dari tempat yang jauh). Untuk memasang paket kita dapat menggunakan parameter aksi **`-i`** atau **`--install`**. Parameter aksi **`-i`** atau **`--install`** ini juga dapat kita gunakan untuk memasang ulang paket, menaikkan (*upgrade*) dan menurunkan (*downgrade*) paket. Pola perintah yang digunakan adalah sebagai berikut:

```
#  dpkg -i <berkas_paket>
```

Sebagai contoh, ketika kita ingin memasang berkas paket perkakas **dpkg** ke dalam sistem, kita dapat menggunakan perintah sebagaimana berikut:

```
#  dpkg -i dpkg_1.18.10-1_amd64.deb
```

Proses pemasangan paket ini terdiri atas beberapa langkah, meliputi:

1) Mengekstraksi berkas-berkas **control** dari paket yang baru.
2) Jika terdapat paket yang sama dengan versi lain telah terpasang sebelum dilakukannya pemasangan baru, maka akan dilakukan eksekusi *script* **prerm** dari paket yang lama (jika tersedia).
3) Menjalankan *script* **preinst** (jika disediakan oleh paket).
4) Membuka (*unpack*) berkas-berkas paket yang yang baru dan pada saat yang sama mencadangkan (*backup*) berkas-berkas milik paket yang lama, sehingga jika terjadi kesalahan, maka berkas-berkas tersebut dapat dipulihkan kembali.
5) Jika terdapat paket yang sama dengan versi lain telah terpasang sebelum dilakukannya pemasangan baru, maka akan dilakukan eksekusi *script* **postrm** dari paket yang lama (jika tersedia). Sebagai catatan, *script* ini dieksekusi setelah *script* **postinst** paket yang baru. Hal ini dikarenakan berkas-berkas yang baru akan ditulis pada saat yang sama dengan penghapusan berkas-berkas yang lama.
6) Mengkonfigurasi paket. Kegiatan konfigurasi paket ini terdiri atas beberapa langkah, meliputi:
   a) Membuka (*unpack*) berkas-berkas konfigurasi (yang terdaftar dalam berkas **conffiles**) dan pada saat yang sama mencadangkan (*backup*) berkas-berkas konfigurasi yang lama, sehingga jika terjadi kesalahan, maka berkas-berkas tersebut dapat dipulihkan kembali.
   b) Menjalankan *script* **postinst** (jika disediakan oleh paket).

Terminal - idnux@debian: ~/Downloads

File Edit View Terminal Tabs Help

```
root@debian:/home/idnux/Downloads# dpkg -i chmsee_1.3.0-2_amd64.deb
Selecting previously unselected package chmsee.
(Reading database ... 92070 files and directories currently installed.)
Preparing to unpack chmsee_1.3.0-2_amd64.deb ...
Unpacking chmsee (1.3.0-2) ...
Setting up chmsee (1.3.0-2) ...
Processing triggers for desktop-file-utils (0.22-1) ...
Processing triggers for mime-support (3.58) ...
Processing triggers for hicolor-icon-theme (0.13-1) ...
Processing triggers for man-db (2.7.0.2-5) ...
root@debian:/home/idnux/Downloads#
```

Gambar 4.6: Pemasangan Paket menggunakan dpkg

Perkakas **dpkg** juga mendukung pemasangan banyak berkas paket yang terdapat di dalam sebuah direktori secara serempak dalam satu baris perintah. Jika yang ingin kita pasang hanya beberapa paket saja (tidak semua berkas paket dalam direktori), maka kita dapat menyebutkan nama berkas paket tersebut secara berurutan. Sebagai contoh, ketika kita ingin memasang berkas paket perkakas **dpkg** dan **libdpkg** ke dalam sistem dalam satu baris perintah, kita dapat menggunakan perintah sebagaimana berikut:

```
#  dpkg -i dpkg_0.18.10-1_amd64.deb libdpkg_0.18.10-1_amd64.deb
```

Dan jika kita ingin memasang semua berkas paket yang terdapat di dalam direktori kerja saat ini, maka kita dapat memanfaatkan karakter *wildcard asteriks* (*****). Contoh:

```
#  dpkg -i *.deb
```

Sedangkan jika kita ingin memasang semua berkas paket yang terdapat di dalam sub-direktori di bawah direktori kerja kita saat ini atau di dalam direktori yang kita tentukan, maka kita dapat memberikan opsi **`-R`** atau **`--recursive`**. Pola perintahnya adalah sebagai berikut:

```
#  dpkg -i -R <nama direktori>
```

Jika diperlukan, kita juga dapat memecah proses pemasangan ini menjadi beberapa bagian, yaitu dengan menjalankan proses pembukaan (*unpack*) dan konfigurasi paket dengan perintah tersendiri. Untuk melakukan kegiatan pembukaan (*unpack*) paket ini, kita dapat menggunakan parameter aksi **`--unpack`**. Pola perintah yang digunakan adalah sebagai berikut:

```
#  dpkg --unpack <berkas paket>
```

Seperti halnya pada kegiatan pemasangan, dalam kegiatan pembukaan (*un-pack*) ini kita juga dapat melakukan pembukaan semua berkas paket yang terdapat di dalam sub-direktori di bawah direktori kerja kita saat ini atau di dalam direktori yang kita tentukan dengan memberikan opsi **`-R`** atau **`--recursive`**. Paket yang telah mengalami proses pembukaan akan memiliki status keadaan paket (*package state*) **unpacked** dalam basis data **dpkg**.

Setelah dibuka, agar paket terpasang secara sempurna maka perlu dilakukan konfigurasi terhadap paket. Untuk melakukan kegiatan konfigurasi paket ini, kita dapat menggunakan parameter aksi **`--configure`**. Pola perintah yang digunakan adalah sebagai berikut:

```
#  dpkg -configure <nama paket>|-a|--pending
```

Untuk mengkorfigurasi banyak paket, kita dapat menyebutkan nama-nama paket tersebut secara berurutan atau kita dapat juga memberikan opsi **`-a`** atau **`--pending`** tanpa perlu menyebutkan nama-nama paket tersebut. Opsi ini akan sangat berguna ketika menemukan banyak paket di dalam sistem yang sudah dibuka namun belum dilakukan konfigurasi. Hal ini bisa terjadi karena kita pernah melakukan pembukaan banyak paket tanpa mengkonfigurasinya (dengan menggunakan parameter aksi **`--unpack`**) atau karena suatu keadaan proses **dpkg** telah terhenti namun kegiatan pemasangan paket belum selesai secara sempurna. Terhentinya proses **dpkg** ini bisa dikarenakan komputer mati mendadak karena adanya pemutusan aliran listrik, adanya interupsi terhadap proses dari perkakas **dpkg** baik dengan sengaja atau tanpa sengaja, atau karena sebab yang lain.

Perkakas **dpkg** ini juga dapat kita gunakan hanya untuk memproses pemicu (*trigger*) yang terdapat di dalam paket. Untuk melakukannya, kita dapat menggunakan parameter aksi **`--triggers-only`**. Opsi ini telah didukung sejak rilis **dpkg 1.14.17**. Pola perintah yang digunakan adalah sebagai berikut:

```
#  dpkg --triggers-only <nama paket>|-a|--pending
```

Parameter aksi ini jika digunakan akan memproses pemicu (*trigger*) yang tertunda, yang ditandai dengan paket akan memiliki status keadaan paket (*package state*) **triggers-pending**. Akan tetapi, penggunaan parameter aksi ini mungkin akan membiarkan paket-paket yang berada dalam keadaan **triggers-awaited** dan **triggers-pending** yang tidak tepat. Masalah ini akan dapat diselesaikan dengan menjalankan perintah **`dpkg --configure --pending`**.

4.3.3. Menghapus Paket

Selain pemasangan paket, salah satu kegiatan utama dalam kegiatan manajemen paket dalam sebuah sistem adalah kegiatan penghapusan paket. Jika dapat kita perbandingkan maka kegiatan penghapusan ini merupakan lawan dari kegiatan pemasangan. Secara teori, penghapusan paket ini merupakan kegiatan penghapusan semua berkas yang berasal dari paket yang sebelumnya dipasang atau ditempatkan ke dalam sistem berkas milik sistem operasi. Di dalam **Sistem Manajemen Paket Debian**, kegiatan penghapusan paket ini terbagi atas dua macam cara, yaitu menghapus (*remove*) dan membersihkan (*purge*).

4.3.3.1. Menghapus (Remove) Paket

Di dalam **Sistem Manajemen Paket Debian**, menghapus (*remove*) paket berarti menghapus paket yang telah terpasang. Kegiatan penghapusan ini akan menghapus semua berkas atau semua hal yang berasal dari paket kecuali berkas-berkas konfigurasi (berkas-berkas konfigurasi yang terdaftar dalam berkas **conffiles**), yang nantinya dapat menghindari dilakukannya konfigurasi ulang paket jika suatu saat nanti kita ingin memasang kembali paket tersebut. Untuk menghapus (*remove*) paket menggunakan **dpkg**, kita dapat menggunakan parameter aksi **`-r`** atau **`--remove`**. Pola perintah yang digunakan adalah sebagai berikut:

```
#  dpkg -r <nama paket>|-a|--pending
```

Sebagai contoh, ketika kita ingin menghapus paket **chmsee** dari sistem, kita dapat menggunakan perintah sebagaimana berikut:

```
#  dpkg -r chmsee
```

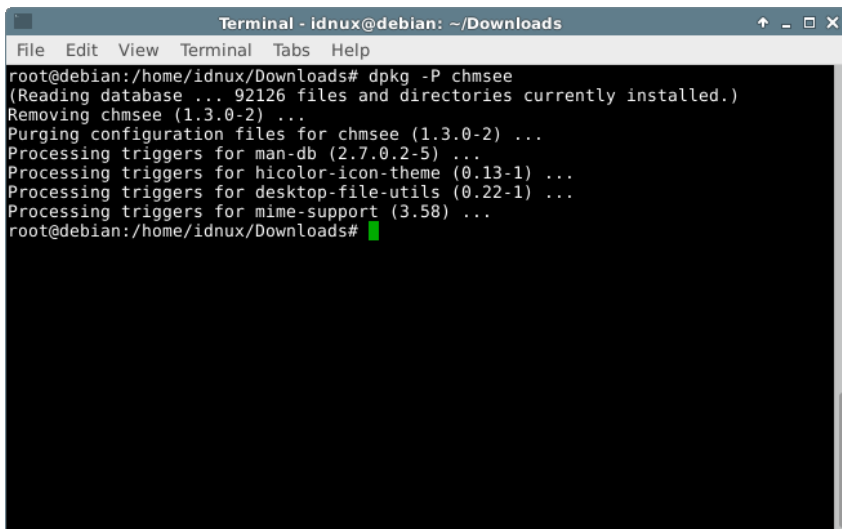
Jika kita memberikan opsi **`-a`** atau **`--pending`** tanpa menyebutkan sebuah nama paket, maka semua paket yang telah ditandai untuk dihapus dalam berkas **/var/lib/dpkg/status**, akan dihapus dari sistem. Paket yang telah dihapus menggunakan parameter aksi ini akan menampilkan keadaannya di dalam kolom (*field*) **Status** yang ada di dalam berkas **/var/lib/dpkg/status** sebagai **non-installed** (biasanya jika paket tidak memiliki berkas konfigurasi) atau **de-install ok config-files** (biasanya jika paket memiliki berkas-berkas konfigurasi yang terdaftar dalam berkas **conffiles**).

```
Terminal - idnux@debian: ~/Downloads
File  Edit  View  Terminal  Tabs  Help
root@debian:/home/idnux/Downloads# dpkg -r chmsee
(Reading database ... 92126 files and directories currently installed.)
Removing chmsee (1.3.0-2) ...
Processing triggers for man-db (2.7.0.2-5) ...
Processing triggers for hicolor-icon-theme (0.13-1) ...
Processing triggers for desktop-file-utils (0.22-1) ...
Processing triggers for mime-support (3.58) ...
root@debian:/home/idnux/Downloads# 
```

Gambar 4.7: Penghapusan Paket menggunakan dpkg

Kegiatan penghapusan paket ini terdiri atas beberapa langkah, yaitu:

1) Menjalankan *script* **prerm**.
2) Menghapus berkas-berkas yang terpasang (kecuali berkas-berkas konfigurasi).
3) Menjalankan *script* **postrm**.

4.3.3.2. Membersihkan (Purge) Paket

Kegiatan membersihkan (*purge*) paket ini berarti membersihkan paket yang telah terpasang atau yang telah dihapus (yang sebelumnya telah dilakukan penghapusan menggunakan parameter aksi **`-r`** atau **`--remove`**). Kegiatan pembersihan ini akan menghapus semua berkas atau semua hal yang berasal dari paket termasuk berkas-berkas konfigurasi. Untuk membersihkan (*purge*) paket ini, kita dapat menggunakan parameter aksi **`-P`** atau **`--purge`**. Pola perintah yang digunakan adalah sebagai berikut:

```
#  dpkg -P <nama paket>|-a|--pending
```

Sebagai contoh, ketika kita ingin membersihkan paket **chmsee** dari sistem, kita dapat menggunakan perintah sebagaimana berikut:

```
#  dpkg -P chmsee
```

Jika kita memberikan opsi **`-a`** atau **`--pending`** tanpa menyebutkan sebuah nama paket, maka semua paket yang telah ditandai untuk dibersihkan (termasuk paket yang telah dihapus menggunakan parameter aksi **`-r`** atau **`--remove`**) dalam berkas **/var/lib/dpkg/status**, akan dibersihkan (*purge*) dari sistem.

Gambar 4.8: Pembersihan Paket menggunakan dpkg

Kegiatan pembersihan paket ini terdiri atas beberapa langkah, yaitu:

1) Menghapus paket. Kegiatan ini dilakukan jika paket belum pernah dilakukan proses penghapusan (menggunakan parameter aksi **`-r`** atau **`--remove`**). Jika paket sebelumnya telah dilakukan proses penghapusan maka **dpkg** akan menghapus berkas-berkas konfigurasi paket.
2) Menjalankan *script* **postrm**.

Beberapa berkas konfigurasi paket mungkin tidak dikenali oleh **dpkg**. Hal ini dikarenakan berkas-berkas konfigurasi tersebut dibuat dan ditangani secara terpisah, seperti melalui *script-script* konfigurasi. Maka dalam kasus ini, **dpkg** tidak akan dapat menghapus berkas konfigurasi tersebut, tetapi penghapusannya

saat proses pembersihan (*purge*) akan diambil alih oleh *script* **postrm** milik paket.

4.3.4. Memeriksa Berkas Paket

dpkg sebagai perkakas manajer paket, selain dibekali dengan kemampuan untuk memasang dan menghapus paket juga menyediakan fungsi untuk melakukan pemeriksaan (inspeksi) terhadap berkas paket. Fungsi pemeriksaan berkas paket ini meliputi verifikasi dan audit (pengujian) terhadap paket, serta perbandingan (komparasi) nomor versi.

4.3.4.1. Verifikasi Paket

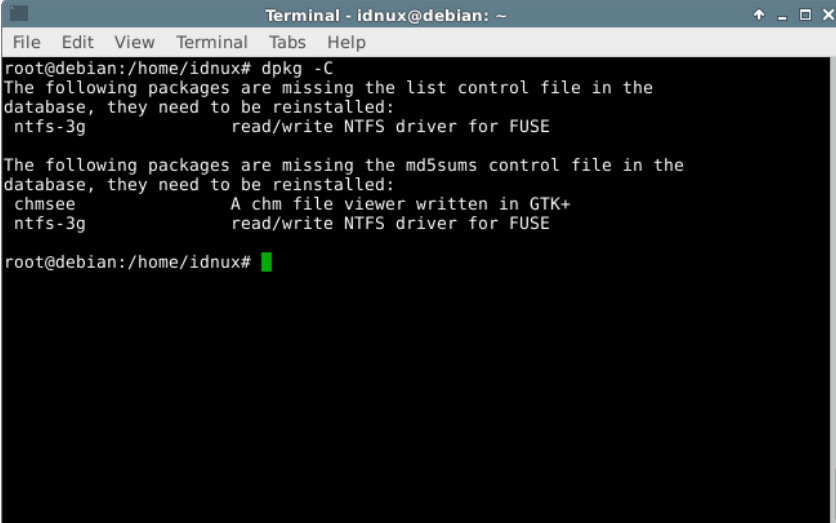
Kegiatan verifikasi keutuhan paket ini dilakukan dengan membandingkan informasi dari berkas-berkas paket yang terpasang dengan informasi dari berkas-berkas meta-data yang tersimpan dalam basis data **dpkg**. Untuk melakukan verifikasi paket ini, kita dapat menggunakan parameter aksi **`-V`** atau **`--verify`**. Pola perintah yang digunakan adalah sebagai berikut:

```
$  dpkg -V <nama paket>
```

Saat menggunakan parameter aksi **`-V`** atau **`--verify`** ini, kita dapat menghilangkan penulisan nama paket. Jika nama paket kita hilangkan, maka **dpkg** akan melakukan verifikasi terhadap semua paket yang terpasang. Untuk saat ini, pemeriksaan fungsional yang dilakukan hanya verifikasi **md5sum** dari isi berkas paket terhadap suatu nilai yang tersimpan dalam berkas basis data. Pengecekan ini akan dilakukan jika basis data berisi berkas **md5sum**. Hasil keluaran dari proses verifikasi ini ditampilkan dalam format **rpm** (saat ini hanya format ini yang didukung), yang terdiri dari satu baris untuk setiap *path* yang gagal saat diperiksa. Baris tersebut dimulai dengan sembilan buah karakter yang melaporkan setiap hasil pemeriksaan secara khusus. Karakter-karakter di dalam baris tersebut biasanya berupa:

- karakter “**?**” menyiratkan bahwa pemeriksaan tidak dapat dilakukan (kurang dukungan, perizinan atau *permission* dan lain sebagainya);
- karakter “**.**”menyiratkan bahwa pemeriksaan telah lulus; dan
- karakter alfanumerik (bisa berupa angka atau huruf) menyiratkan kegagalan pemeriksaan secara khusus. Kegagalan verifikasi **md5sum** (seperti karena isi berkas telah diubah) dilambangkan dengan angka lima (**5**) pada karakter ketiga.

Selain itu, baris tersebut juga diikuti dengan sebuah spasi dan sebuah karakter atribut atau sifat (untuk saat ini huruf “**c**” menunjukkan berkas **conffiles**), spasi lagi dan nama *path*.

```
Terminal - idnux@debian: ~
File  Edit  View  Terminal  Tabs  Help
??5??????   /usr/share/locale/da/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/sk/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/pt_BR/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/hr/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/cs/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/ug/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/ja/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/pa/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/pt/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/es/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/it/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/lv/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/eo/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/gl/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/ur_PK/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/nb/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/pl/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/sq/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/fi/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/ru/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/hu/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/ko/LC_MESSAGES/xfce4-netload-plugin.mo
??5??????   /usr/share/locale/lt/LC_MESSAGES/xfce4-netload-plugin.mo
root@debian:/home/idnux#
```

Gambar 4.9: Contoh Hasil Keluaran Verifikasi Paket

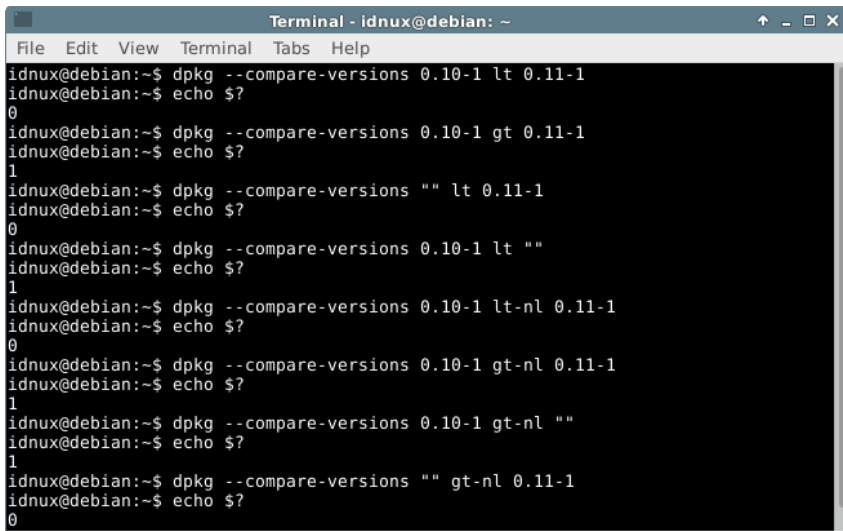
4.3.4.2. Audit (Pengujian) Paket

Kegiatan audit atau pengujian paket ini dilakukan dengan melakukan pemeriksaan stabilitas atau kenormalan dan konsistensi basis data dari suatu paket tertentu. Sebagai contoh, mencari paket yang terpasang hanya sebagian di dalam sistem atau yang telah hilang, atau paket yang memiliki berkas atau data **control** yang salah atau usang (tidak dipakai lagi). Untuk melakukan kegiatan audit paket ini, kita dapat menggunakan parameter aksi **-C** atau **--audit**. Pola perintah yang digunakan adalah sebagai berikut:

```
$  dpkg -C <nama paket>
```

Saat menggunakan parameter aksi **`-C`** atau **`--audit`** ini, kita dapat menghilangkan penulisan nama paket. Jika nama paket kita hilangkan, maka **dpkg** akan melakukan audit terhadap semua paket yang terpasang. Hasil dari kegiatan audit ini adalah **dpkg** akan memberikan saran tentang apa yang harus dilakukan untuk memperbaiki masalah yang disebutkan.

```
root@debian:/home/idnux# dpkg -C
The following packages are missing the list control file in the
database, they need to be reinstalled:
 ntfs-3g              read/write NTFS driver for FUSE

The following packages are missing the md5sums control file in the
database, they need to be reinstalled:
 chmsee               A chm file viewer written in GTK+
 ntfs-3g              read/write NTFS driver for FUSE

root@debian:/home/idnux#
```

Gambar 4.10: Contoh Hasil Keluaran Audit Paket

4.3.4.3. Perbandingan Nomor Versi

Sebagai perkakas manajer paket, **dpkg** juga menyediakan fungsi untuk melakukan perbandingan nomor versi paket. Untuk melakukan perbandingan nomor versi ini, kita dapat menggunakan parameter aksi **`--compare-versions`**. Pola perintah yang digunakan adalah sebagai berikut:

```
$ dpkg --compare-versions versi1 operator versi2
```

Dalam perintah ini, operator yang dapat digunakan terdiri atas dua kelompok operator yang memiliki perbedaan dalam cara memperlakukan sebuah versi kosong apakah yang pertama atau kedua. Kedua kelompok operator tersebut adalah:

1) Kelompok operator pertama memperlakukan sebuah versi kosong sebagai yang lebih dahulu dibandingkan suatu versi tertentu. Operator-operator yang termasuk dalam kelompok pertama ini yaitu **lt** (*less than* “**<<**”), **le** (*less than or equal to* “**<=**”), **eq** (*equal to* “**=**”), **ne** (*not equal* “**≠**”), **ge** (*greater than or equal to* “**>=**”) dan **gt** (*greater than* “**>>**”).
2) Kelompok operator kedua memperlakukan sebuah versi kosong sebagai yang lebih belakangan dari suatu versi tertentu. Operator-operator yang termasuk dalam kelompok kedua ini yaitu **lt-nl** (*less than* “**<<**”), **le-nl** (*less than or equal to* “**<=**”), **ge-nl** (*greater than or equal to* “**>=**”) dan **gt-nl** (*greater than* “**>>**”).

```
Terminal - idnux@debian: ~
File  Edit  View  Terminal  Tabs  Help
idnux@debian:~$ dpkg --compare-versions 0.10-1 lt 0.11-1
idnux@debian:~$ echo $?
0
idnux@debian:~$ dpkg --compare-versions 0.10-1 gt 0.11-1
idnux@debian:~$ echo $?
1
idnux@debian:~$ dpkg --compare-versions "" lt 0.11-1
idnux@debian:~$ echo $?
0
idnux@debian:~$ dpkg --compare-versions 0.10-1 lt ""
idnux@debian:~$ echo $?
1
idnux@debian:~$ dpkg --compare-versions 0.10-1 lt-nl 0.11-1
idnux@debian:~$ echo $?
0
idnux@debian:~$ dpkg --compare-versions 0.10-1 gt-nl 0.11-1
idnux@debian:~$ echo $?
1
idnux@debian:~$ dpkg --compare-versions 0.10-1 gt-nl ""
idnux@debian:~$ echo $?
1
idnux@debian:~$ dpkg --compare-versions "" gt-nl 0.11-1
idnux@debian:~$ echo $?
0
```

Gambar 4.11: Contoh penggunaan parameter aksi --compare-versions

4.3.5. Dukungan terhadap Banyak Arsitektur (Multi-Arch)

Multi-arch adalah istilah yang digunakan untuk merujuk pada kemampuan suatu sistem untuk memasang dan menjalankan aplikasi dari berbagai target biner yang berbeda dalam sistem yang sama. Sebagai contoh, menjalankan sebuah aplikasi **i386-linux-gnu** (**x86** atau **IA32**) pada sistem **amd64-linux-gnu** (**x86_64**). Perkakas **dpkg** telah mendukung banyak arsitektur (*multi architecture*) atau **multi-arch** ini sejak versi **1.16.2** yang dirilis pada tanggal 19 Maret 2012.

Semua paket **Debian** memiliki kolom (*field*) **Architecture** di dalam berkas **control-**nya. Kolom ini dapat berisi "**all**" (untuk paket *architecture independent*) atau atau nama arsitektur yang menjadi target paket (seperti **amd64**, **i386**, **aarch64** dan lain sebagainya). Dalam kondisi aslinya (*default*), **dpkg** hanya akan dapat memasang jika arsitekturnya cocok atau sesuai dengan arsitektur sistem. Hal ini untuk memastikan pengguna tidak akan menjalankan berkas biner atau aplikasi yang dikompilasi untuk arsitektur yang tidak sesuai. Namun hal ini diberikan pengecualian, terutama bagi komputer yang dapat menjalankan berkas biner atau aplikasi dari berbagai arsitektur, baik secara bawaan (*native*) seperti sistem **x86_64** yang dapat menjalankan berkas biner **IA-32** (32 bit) melalui mode kompatibilitas (*compatibility mode*), atau melalui **emulator**.

Untuk mengetahui arsitektur yang digunakan oleh sistem saat ini, kita dapat menggunakan parameter aksi **`--print-architecture`**. Pola perintah yang digunakan adalah sebagai berikut:

```
$  dpkg --print-architecture
```

Sistem operasi komputer modern yang ada saat ini biasanya dibuat untuk mendukung dan dapat berjalan pada satu atau lebih macam arsitektur. Begitu juga dengan **Debian**, saat ini sistem operasi **Debian** telah mendukung banyak macam arsitektur, yang terdiri atas:

- 10 arsitektur dengan dukungan dan dirilis secara resmi;
- 3 arsitektur telah digantikan oleh arsitektur yang lain;
- 9 arsitektur sedang dalam pengerjaan;
- 4 arsitektur telah dihentikan dukungannya; dan
- 4 arsitektur telah benar-benar mati (proyeknya).

Berikut ini adalah daftar arsitektur yang telah didukung oleh **Debian** sampai dengan saat ini:

Tabel 4.4: Daftar Arsitektur yang didukung oleh Sistem Operasi Debian

No.	**Nama Arsitektur**	*Jenis Arsitektur Prosesor*	*Status Dukungan*	*Keterangan*
1	**i386**	IA32 atau 32 bit PC	Didukung Resmi	Versi 32 bit dari keluarga prosesor **x86**. Merupakan arsitektur pertama yang didukung oleh sistem operasi **Debian**.
2	**amd64**	X86_64 atau 64 bit PC	Didukung Resmi	Versi 64 bit dari keluarga prosesor **x86**. Didukung secara resmi sejak rilis **Debian 4.0**.
3	**powerpc**	Motorola/IBM PowerPC	Didukung Resmi	Merupakan arsitektur microprocessor **RISC** yang dikembangkan oleh **IBM**, **Motorola** (sekarang **Freescale**) dan **Apple**. Arsitektur ini mengizinkan kedua implementasi 64 bit dan 32 bit. Didukung secara resmi sejak rilis **Debian 2.2**.
4	**armel**	EABI ARM	Didukung Resmi	Prosesor 32 bit **ARM** *little-endian* yang kompatibel dengan set atau kumpulan instruksi **ARMv4t**, **ARM-v5t** dan di atasnya. Didukung secara resmi sejak rilis **Debian 5.0**.
5	**armhf**	Hard Float ABI ARM	Didukung Resmi	Prosesor 32 bit **ARM** *little-endian* yang kompatibel dengan set atau kumpulan instruksi **ARMv7** termasuk tambahan perangkat keras *floating-point* **VFP3-D16** tambahan (dan **Thumb-2**) dan di atasnya. Didukung secara resmi sejak rilis **Debian 7.0**.
6	**mips**	MIPS (big-endian)	Didukung Resmi	Prosesor arsitektur **MIPS** dengan mode *big-endian.* Didukung secara resmi sejak rilis **Debian 3.0**.
7	**mipsel**	MIPS (little-endian)	Didukung Resmi	Prosesor arsitektur **MIPS** dengan mode *little-endian.* Didukung secara resmi sejak rilis **Debian 3.0**.
8	**arm64**	64 bit ARM (AArch64)	Didukung Resmi	Prosesor 64 bit **ARM** yang kompatibel dengan set atau kumpulan instruksi **ARMv8**. Didukung secara resmi sejak rilis **Debian 8.0**.
9	**ppc64el**	POWER7+, POWER8	Didukung Resmi	Prosesor PowerPC 64 bit mode *little-endian* yang menggunakan **ABI Open Power**

No.	**Nama Arsitektur**	*Jenis Arsitektur Prosesor*	*Status Dukungan*	*Keterangan*
				ELFv2, seperti prosesor **POWER8**. Didukung secara resmi sejak rilis **Debian 8.0**.
10	**s390x**	System z	Didukung Resmi	*Userland* 64 bit untuk komputer mainframe **IBM System z**. Didukung secara resmi sejak rilis **Debian 7.0**.
11	**arm**	OABI ARM	digantikan oleh **armel**	Prosesor 32 bit **ARM**. Didukung sejak **Debian 2.2** sampai **Debian 5.0**.
12	**s390**	S/390 and zSeries	digantikan oleh **s390x**	Digunakan untuk komputer *server* dan *mainframe* **IBM S/390** dan **IBM zSeries**. Didukung sejak **Debian 3.0** sampai **Debian 7.0**. Sejak rilis **Debian 8.0** digantikan oleh **s390x**.
13	**sparc**	Sun SPARC	digantikan oleh **sparc64**	Merupakan arsitektur microprocessor **RISC** yang dikembangkan oleh **Sun Microsystems**. Didukung sejak rilis **Debian 2.1** sampai **Debian 7.0**. Sejak **Debian 8.0** tidak lagi didukung dan rencananya akan digantikan oleh **sparc64**.
14	**x32**	64 bit PC dengan pointer 32 bit	dalam pengerjaan	**X32** adalah sebuah **ABI** untuk prosesor **x86_64** yang menggunakan *integer* dan *pointer* 32 bit.
15	**m68k**	Motorola 68k	dalam pengerjaan	Prosesor **Motorola** seri **68k**. Didukung sejak **Debian 2.0** sampai **Debian 3.1**. Saat ini sedang dalam usaha untuk dibangkitkan lagi.
16	**mips64el**	MIPS (64-bit little-endian mode)	dalam pengerjaan	Prosesor arsitektur **MIPS** 64 bit dengan mode *little-endian* yang menggunakan set atau kumpulan instruksi **MIPS64r2**, **ABI N64** dan perangkat keras *floating-point*.
17	**sparc64**	64-bit SPARC	dalam pengerjaan	Versi 64 bit dari prosesor **SPARC**.
18	**sh4**	SuperH	dalam pengerjaan	**SuperH** adalah arsitektur prosesor **RISC** yang awalnya dikembangkan oleh **Hitachi** dan saat ini dibuat oleh **Renesas Electronics**. **SH4** adalah salah satu varian 32 bit dari arsitektur **SuperH**.
19	**powerpcspe**	PowerPC Signal Processing Engine	dalam pengerjaan	Prosesor 32 bit daya rendah **PowerPC e500** dari **FreeScale** dan **IBM** dengan perangkat keras *Signal Processing Engine*.
20	**hurd-i386**	IA32 atau 32 bit PC	dalam pengerjaan	Versi 32 bit dari keluarga prosesor **x86** dengan kernel **GNU Hurd**.
21	**kfreebsd-i386**	IA32 atau 32 bit PC	dalam pengerjaan	Versi 32 bit dari keluarga prosesor **x86** dengan *userland* **GNU** dan kernel **FreeBSD**. Dirilis pertama kali pada **Debian 6.0** seba-

No.	**Nama Arsitektur**	*Jenis Arsitektur Prosesor*	*Status Dukungan*	*Keterangan*
				gai *technology preview* dan dirilis sebagai rilis resmi pada **Debian 7.0**. Dalam rilis **Debian 8.0** tidak lagi termasuk dalam rilis resmi, namun masih tetap dikembangkan.
22	**kfreebsd-amd64**	X86_64 atau 64 bit PC	dalam pengerjaan	Versi 64 bit dari keluarga prosesor **x86** dengan *userland* **GNU** dan kernel **FreeBSD**. Dirilis pertama kali pada **Debian 6.0** sebagai *technology preview* dan dirilis sebagai rilis resmi pada **Debian 7.0**. Dalam rilis **Debian 8.0** tidak lagi termasuk dalam rilis resmi, namun masih tetap dikembangkan.
23	**alpha**	Alpha	dihentikan	Merupakan arsitektur microprocessor **RISC** 64 bit yang dikembangkan oleh **Digital Equipment Corporation** (**DEC**). Didukung sejak **Debian 2.1** sampai **Debian 5.0**.
24	**AVR32**	Atmel 32-bit RISC	dihentikan	Merupakan arsitektur microprocessor **RISC** 32 bit yang dikembangkan oleh **Atmel Corporation**.
25	**hppa**	HP PA-RISC	dihentikan	Merupakan arsitektur microprocessor **RISC** 64 bit yang dikembangkan oleh **Hewlett-Packard** (**HP**). Didukung sejak **Debian 3.0** sampai **Debian 5.0**.
26	**ia64**	Intel Itanium IA-64	dihentikan	Merupakan arsitektur microprocessor **EPIC** (*Explicitly Parallel Instruction Computing*) 64 bit yang dikembangkan oleh **Intel Corporation** dan **Hewlett-Packard** (**HP**). Didukung sejak **Debian 3.0** sampai **Debian 7.0**.
27	**m32**	M32R	mati	Merupakan arsitektur microprocessor **RISC** 32 bit yang dikembangkan oleh **Renesas Electronics**.
28	**netbsd-i386**	IA32 atau 32 bit PC	mati	Versi 32 bit dari keluarga prosesor **x86** dengan *userland* **GNU** dan kernel **NetBSD**. Tidak pernah dirilis.
29	**netbsd-alpha**	Alpha	mati	**Merupakan arsitektur microprocessor RISC 64 bit yang dikembangkan oleh Digital Equipment Corporation** (**DEC**) dengan *userland* **GNU** dan kernel **NetBSD**. Tidak pernah dirilis
30	**or1k**	OpenRISC 1200	mati	Merupakan arsitektur microprocessor **RISC** 32 bit kode terbuka yang dikembangkan oleh komunitas **OpenCores**.

Agar kita dapat memasang paket yang ditujukan untuk arsitektur yang berbeda dengan arsitektur sistem yang kita gunakan saat ini, maka kita perlu untuk menambahkan atau mendaftarkan arsitektur tambahan atau yang lebih dikenal dengan sebutan arsitektur asing (*foreign architecture*) tersebut ke dalam basis

data arsitektur **dpkg**. Untuk menambahkan arsitektur asing ini, kita dapat menggunakan parameter aksi **`--add-architecture`**. Pola perintah yang digunakan adalah sebagai berikut:

```
# dpkg --add-architecture <nama arsitektur>
```

Sebagai contoh, ketika kita ingin menambahkan dukungan terhadap arsitektur **IA-32** atau yang di **Debian** lebih dikenal dengan nama **i386** di sistem **x86_64** atau **amd64** yang kita miliki, kita dapat menggunakan perintah sebagaimana berikut:

```
# dpkg --add-architecture i386
```

Arsitektur asing yang telah kita tambahkan tersebut akan disimpan di dalam berkas **/var/lib/dpkg/arch**. Berkas **/var/lib/dpkg/arch** ini mungkin tidak akan kita temukan, jika sistem kita masih belum kita atur agar mendukung arsitektur yang lain. Setelah arsitektur kita tambahkan, kita dapat mengetahui arsitektur asing apa saja yang telah terdaftar dengan menggunakan parameter aksi **`--print-foreign-architectures`**. Pola perintah yang digunakan adalah sebagai berikut:

```
$ dpkg --print-foreign-architectures
```

Dan jika kita ingin menghapus atau menghilangkan dukungan untuk suatu arsitektur asing (*foreign architecture*), kita dapat menggunakan parameter aksi **`--remove-architecture`**. Pola perintah yang digunakan adalah sebagai berikut:

```
# dpkg --remove-architecture <nama arsitektur>
```

Setelah kita mendaftarkan arsitektur asing ke basis data arsitektur **dpkg**, artinya saat ini kita telah diizinkan untuk memasang paket yang ditujukan untuk arsitektur yang telah kita daftarkan tersebut. Untuk memasang paket, kita dapat menggunakan pola perintah seperti biasanya. Hanya saja, kita harus memastikan bahwa paket yang akan kita pasang telah sesuai dengan arsitektur asing tersebut. Sedangkan untuk menghapus paket arsitektur asing atau kegiatan pengelolaan paket lain, kita harus menambahkan karakter **colon** ("**:**") dan nama arsitektur di belakang nama paket. Pola perintah yang harus kita gunakan adalah sebagai berikut:

```
# dpkg -r <nama paket>:<nama arsitektur>
```

Sebagai contoh, ketika kita ingin menghapus paket **nano** yang merupakan paket dengan arsitektur asing **i386** dari sistem, maka kita dapat menggunakan perintah sebagaimana berikut:

```
# dpkg -r nano:i386
```

```
Terminal - idnux@debian: ~
File  Edit  View  Terminal  Tabs  Help
idnux@debian: ~                                    idnux@debian: ~
root@debian:/home/idnux/Downloads# dpkg --print-architecture
amd64
root@debian:/home/idnux/Downloads# dpkg --add-architecture i386
root@debian:/home/idnux/Downloads# dpkg --print-foreign-architectures
i386
root@debian:/home/idnux/Downloads# dpkg -i nano_2.2.6-3_i386.deb libncursesw5_5.
9+20140913-1+b1_i386.deb libtinfo5_5.9+20140913-1+b1_i386.deb
Selecting previously unselected package nano.
(Reading database ... 93242 files and directories currently installed.)
Preparing to unpack nano_2.2.6-3_i386.deb ...
Unpacking nano (2.2.6-3) ...
Selecting previously unselected package libncursesw5:i386.
Preparing to unpack libncursesw5_5.9+20140913-1+b1_i386.deb ...
Unpacking libncursesw5:i386 (5.9+20140913-1+b1) ...
Selecting previously unselected package libtinfo5:i386.
Preparing to unpack libtinfo5_5.9+20140913-1+b1_i386.deb ...
Unpacking libtinfo5:i386 (5.9+20140913-1+b1) ...
Setting up libtinfo5:i386 (5.9+20140913-1+b1) ...
Setting up libncursesw5:i386 (5.9+20140913-1+b1) ...
Setting up nano (2.2.6-3) ...
update-alternatives: using /bin/nano to provide /usr/bin/editor (editor) in auto
 mode
update-alternatives: using /bin/nano to provide /usr/bin/pico (pico) in auto mod
e
```

Gambar 4.12: Mencoba memasang Paket dari Arsitektur Lain

4.3.6. Status Keluaran Perintah dpkg

Pada saat setelah menyelesaikan kerja atau aksinya, perkakas **dpkg** akan memberikan sebuah status keluaran (*exit status*) atau *return code*. Status keluaran ini mengindikasikan tentang keberhasilan kerja dari perkakas **dpkg** ini dan juga dapat digunakan sebagai petunjuk awal saat terjadi kegagalan. Dengan ditemukannya petunjuk awal ini, maka diharapkan akan dapat membantu pengguna dalam melakukan analisa tentang penyebab kegagalan tersebut dan langkah perbaikan apa saja yang harus dilakukan. Berikut ini adalah daftar status keluaran dari perintah **dpkg**:

Tabel 4.5: Daftar Status Keluaran Perintah dpkg

No.	*Status Keluaran*	**Arti**
1	0	Aksi atau tindakan yang diminta telah berhasil dilaksanakan. Atau perintah pemeriksaan atau penegasan menghasilkan nilai “benar” (*true*).
2	1	Perintah pemeriksaan atau penegasan menghasilkan “salah” (*false*).
3	2	Kesalahan fatal atau tidak dapat dipulihkan karena penggunaan baris perintah yang salah atau karena berhubungan dengan sistem, seperti akses ke basis data, alokasi memori dan lain-lain.

4.4. Perkakas-perkakas Lain di dalam Paket dpkg

4.4.1. dpkg-deb

Perkakas **dpkg-deb** merupakan perkakas manipulasi berkas paket (biner) **Debian**. Perkakas ini dapat digunakan untuk memaketkan, membuka dan menyediakan informasi tentang berkas paket (biner) **Debian**. Perkakas ini merupakan perkakas satu-satunya yang dapat mengenali format tersebut. Perkakas ini pada awalnya menggunakan nama **dpkg-util.deb**. Kemudian pada tanggal 3 Oktober 1994, **dpkg-util.deb** diubah namanya menjadi **dpkg-deb** dan nama ini tetap digunakan sampai dengan sekarang.

Perkakas **dpkg-deb** ini dikendalikan dengan menggunakan perintah (*command*) tertentu dan tambahan sejumlah opsi (*option*) jika diperlukan. Pola perintah yang digunakan adalah sebagai berikut:

```
dpkg-deb [<opsi>...] <perintah>
```

Perkakas **dpkg-deb** ini juga dapat dijalankan dengan memanggil perintah **dpkg**, seolah-olah perkakas **dpkg** akan mengamati bahwa opsi yang diminta telah sesuai dengan **dpkg-deb** dan memanggil atau menjalankan **dpkg-deb** dengan argumen yang sama. Berikut ini adalah daftar dan penjelasan ringkas dari perintah **dpkg-deb** yang dapat kita gunakan:

Tabel 4.6: Daftar Perintah (Command) dpkg-deb

No.	**Perintah**	*Format Perintah*	*Kegunaan*
1	`-b, --build`	`dpkg-deb -b, --build <direktori> [<berkas paket>|<direktori>]`	Membuat berkas paket **Debian** dari pohon sistem berkas yang tersimpan di dalam sebuah direktori.
2	`-I, --info`	`dpkg-deb -I, --info <berkas paket> [<nama berkas control>...]`	Menyediakan informasi tentang berkas paket biner.
3	`-W, --show`	`dpkg-deb -W, --show <berkas paket>`	Menyediakan informasi tentang berkas paket biner dalam format yang telah ditentukan dengan nilai opsi --**showformat**.
4	`-f, --field`	`dpkg-deb -f, --field <berkas paket> [<nama berkas control>...]`	Menampilkan informasi berkas **control** dari berkas paket biner.
5	`-c, --contents`	`dpkg-deb -c, --contents <berkas paket>`	Menampilkan daftar isi pohon sistem berkas yang merupakan bagian dari berkas paket.
6	`-x, --extract`	`dpkg-deb -x, --extract <berkas paket> <direktori>`	Mengekstrak pohon sistem berkas dari berkas paket ke direktori yang telah ditentukan.

No.	**Perintah**	*Format Perintah*	*Kegunaan*
7	-X, --vextract	dpkg-deb -X, --vextract <berkas paket> <direktori>	Mengekstrak pohon sistem berkas dari berkas paket ke direktori yang telah ditentukan dan menampilkan daftar berkas-berkas yang diekstrak tersebut.
8	-e, --control	dpkg-deb -e, --control <berkas paket> <direktori>	Mengekstrak berkas-berkas **control** dari berkas paket ke direktori yang telah ditentukan.
9	-R, --raw-extract	dpkg-deb -R, --raw-extract <berkas paket> <direktori>	Mengekstrak pohon sistem berkas dari berkas paket ke direktori yang telah ditentukan dan berkas-berkas **control** ke sub-direktori **DEBIAN** dari direktori tersebut.
10	--ctrl-tar-file	dpkg-deb --ctrl-tar-file <berkas paket>	Mengekstrak data **control** dari sebuah paket biner dan mengirimkannya ke dalam keluaran standar dalam format **tar**.
11	--fsys-tar-file	dpkg-deb --fsys-tar-file <berkas paket>	Mengekstrak data pohon sistem berkas dari sebuah paket biner dan mengirimkannya ke dalam keluaran standar dalam format **tar**.
12	-?, --help	dpkg-deb -?, --help	Menampilkan pesan penggunaan.
13	--version	dpkg-deb --version	Menampilkan versi perkakas **dpkg-deb**.

Sedangkan untuk daftar dan penjelasan ringkas dari opsi perintah **dpkg-deb** yang dapat kita gunakan adalah sebagai berikut:

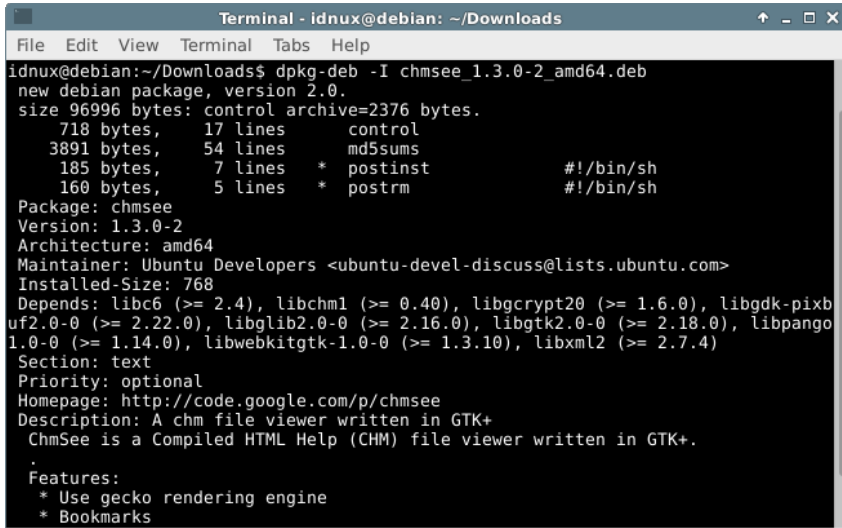
Tabel 4.7: Daftar Opsi Perintah dpkg-deb

No.	**Opsi**	*Kegunaan*
1	--showformat=<format>	Menentukan format keluaran yang dihasilkan dari perintah **--show**. Format bakunya adalah "**${Package}\t${Version}\n**".
2	-z<tingkat kompresi>	Menentukan tingkat kompresi yang digunakan pada program *backend* pengompres saat pembangunan paket. Nilai tingkat kompresi yang diperbolehkan adalah dari 0 sampai 9. Nilai bakunya adalah 9 untuk **gzip** dan 6 untuk **xz.**
3	-s<strategi kompresi>	Menentukan strategi kompresi yang digunakan pada program *backend* pengompres saat pembangunan paket. Nilai yang diperbolehkan adalah **none**, **filtered**, **huffman**, **rle** dan **fixed** untuk **gzip** serta **extreme** untuk **xz**.

No.	Opsi	*Kegunaan*
4	`-z<tipe kompresi>`	Menentukan tipe kompresi yang digunakan pada program *backend* pengompres saat pembangunan paket. Nilai yang diperbolehkan adalah **gzip**, **xz** dan **none**. Nilai bakunya adalah **xz**.
5	`--uniform-compression`	Menentukan bahwa parameter kompresi yang sama akan digunakan untuk semua anggota berkas paket (yaitu berkas **control.tar** dan **data.tar**). Tipe kompresi yang diperbolehkan yang didukung hanya **none**, **gzip** dan **xz**.
6	`--deb-format=<format>`	Mengatur versi format paket biner yang digunakan saat pembangunan paket. Nilai yang diperbolehkan adalah **2.0** (sebagai format baru) dan **0.939000** (format lama). Nilai bakunya adalah **2.0**.
7	`--nocheck`	Mencegah pemeriksaan atas isi dari sebuah paket yang biasa dilakukan saat pembangunan paket.
8	`-v, --verbose`	Mengaktifkan keluaran secara rinci.
9	`-D, --debug`	Mengaktifkan keluaran awakutu (*debugging*).

Sebagaimana **dpkg**, perkakas **dpkg-deb** pada saat setelah menyelesaikan kerja**nya juga** akan memberikan sebuah status keluaran (*exit status*) atau *return code*. Status keluaran ini mengindikasikan tentang keberhasilan kerja dari perkakas **dpkg-deb** ini dan juga dapat digunakan sebagai petunjuk awal saat terjadi kegagalan. Berikut ini adalah daftar status keluaran dari perintah **dpkg-deb**:

Tabel 4.8: Daftar Status Keluaran Perintah dpkg-deb

No.	*Status Keluaran*	**Arti**
1	0	Aksi atau tindakan yang diminta telah berhasil dilaksanakan.
2	2	Kesalahan fatal atau tidak dapat dipulihkan karena penggunaan baris perintah yang salah atau karena berhubungan dengan sistem, seperti akses ke basis data, alokasi memori dan lain-lain.

4.4.1.1. Mendapatkan Informasi tentang Berkas Paket

Penyediaan informasi yang berkaitan dengan suatu berkas paket biner **Debian** merupakan salah satu fungsi atau kegunaan dari perkakas **dpkg-deb** ini. Untuk mendapatkan informasi tentang berkas paket tersebut, kita dapat menggunakan beberapa perintah (*command*). Perintah-perintah tersebut adalah **`-I`** atau **`--info`**, **`-W`** atau **`--show`**, **`-c`** atau **`--contents`** dan **`-f`** atau **`--field`**. Namun, jika kita ingin mendapatkan informasi tentang berkas paket **Debian** secara lengkap, maka perintah yang lebih tepat untuk kita gunakan adalah perintah **`-I`** atau **`--info`**. Pola perintah yang digunakan adalah sebagai berikut:

```
$  dpkg-deb -I <berkas paket> [<nama berkas control>...]
```

Sebagai contoh, ketika kita ingin mendapatkan informasi tentang berkas paket **chmsee_1.3.0-2_amd64.deb**, kita dapat menggunakan perintah sebagaimana berikut:

```
$  dpkg-deb -I chmsee_1.3.0-2_amd64.deb
```

```
Terminal - idnux@debian: ~/Downloads
File  Edit  View  Terminal  Tabs  Help
idnux@debian:~/Downloads$ dpkg-deb -I chmsee_1.3.0-2_amd64.deb
 new debian package, version 2.0.
 size 96996 bytes: control archive=2376 bytes.
     718 bytes,    17 lines      control
    3891 bytes,    54 lines      md5sums
     185 bytes,     7 lines   *  postinst             #!/bin/sh
     160 bytes,     5 lines   *  postrm               #!/bin/sh
 Package: chmsee
 Version: 1.3.0-2
 Architecture: amd64
 Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
 Installed-Size: 768
 Depends: libc6 (>= 2.4), libchm1 (>= 0.40), libgcrypt20 (>= 1.6.0), libgdk-pixb
uf2.0-0 (>= 2.22.0), libglib2.0-0 (>= 2.16.0), libgtk2.0-0 (>= 2.18.0), libpango
1.0-0 (>= 1.14.0), libwebkitgtk-1.0-0 (>= 1.3.10), libxml2 (>= 2.7.4)
 Section: text
 Priority: optional
 Homepage: http://code.google.com/p/chmsee
 Description: A chm file viewer written in GTK+
  ChmSee is a Compiled HTML Help (CHM) file viewer written in GTK+.
  .
  Features:
   * Use gecko rendering engine
   * Bookmarks
```

Gambar 4.13: Contoh keluaran dari perintah dpkg-deb -I

Perintah **`-I`** atau **`--info`** ini akan memberikan beberapa macam informasi tentang berkas paket. Informasi-informasi tersebut meliputi:

- versi format dari berkas paket;
- ukuran berkas paket dan ukuran berkas arsip **control** yang terdapat di dalam paket;
- ukuran berkas dari masing-masing berkas **control** yang terdapat di dalam paket dan jumlah baris yang terdapat pada setiap berkas **control** tersebut; dan
- isi dari berkas **control**.

Selain itu, perintah **`-I`** atau **`--info`** ini juga dapat digunakan untuk hanya menampilkan isi dari berkas-berkas **control** dengan cara menambahkan nama berkas **control** setelah penulisan nama berkas paket. Sebagai contoh, ketika kita ingin melihat isi dari berkas **md5sums** dan berkas **postinst** yang ada di dalam berkas paket **chmsee**, maka kita dapat menggunakan perintah sebagaimana berikut:

```
$ dpkg-deb -I chmsee_1.3.0-2_amd64.deb postinst md5sums
```

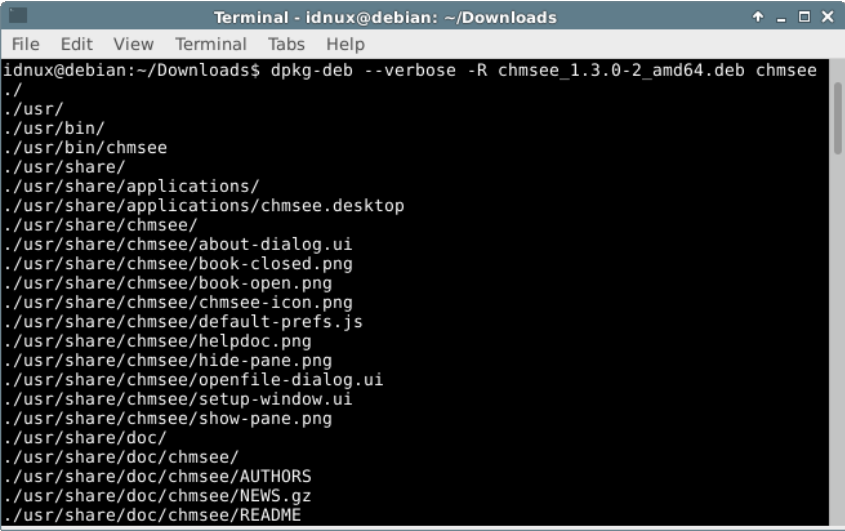
```
Terminal - idnux@debian: ~/Downloads
File  Edit  View  Terminal  Tabs  Help
idnux@debian:~/Downloads$ dpkg-deb -I chmsee_1.3.0-2_amd64.deb postinst md5sums
#!/bin/sh
set -e
# Automatically added by dh_installmenu
if [ "$1" = "configure" ] && [ -x "`which update-menus 2>/dev/null`" ]; then
        update-menus
fi
# End automatically added section
4a2e0a4cd7d0b8f42ea6a388d0cc30e0  usr/bin/chmsee
747b7915ab4b6cde6b4c032f72d3531b  usr/share/applications/chmsee.desktop
5870a1b552097720b5e46e78e4b5af02  usr/share/chmsee/about-dialog.ui
c3cccd5cba706acc673d608e5a91e7a0  usr/share/chmsee/book-closed.png
1a7ac1aa30abdcf1c22bc9e19a5fa51b  usr/share/chmsee/book-open.png
8f75e7f9ec5351d672d63c8ea6dfe9e6  usr/share/chmsee/chmsee-icon.png
c1dab7b0bdaebe61ceed23096e811db3  usr/share/chmsee/default-prefs.js
ef9f97ca90eeffda318fe6b39bd3ed87  usr/share/chmsee/helpdoc.png
c981841f3e96c0d53b500fa41e2b5409  usr/share/chmsee/hide-pane.png
f908abb855ef3232bef3b517a24932d9  usr/share/chmsee/openfile-dialog.ui
8afe68aea9a24da40e04262c66b28888  usr/share/chmsee/setup-window.ui
8250ceb95e3cce675a25e70ff9bd9d3b  usr/share/chmsee/show-pane.png
962f9867d659f7ac8c746000dd52f8dd  usr/share/doc/chmsee/AUTHORS
477cde5662036665f25d61b13f1f1bd0  usr/share/doc/chmsee/NEWS.gz
d483155074fcfed28a62e35871da4a37  usr/share/doc/chmsee/README
82924ef477ef67526690d6eceb52c65c  usr/share/doc/chmsee/changelog.Debian.gz
```

Gambar 4.14: Menampilkan isi berkas postinst dan md5sums menggunakan dpkg-deb

4.4.1.2. Mengekstrak Isi Berkas Paket Debian

Sebagai perkakas manipulasi berkas paket, **dpkg-deb** memiliki berbagai macam fungsi atau kegunaan. Salah satu kegunaan lain dari perkakas ini adalah mengekstrak isi dari berkas paket **Debian**. Untuk mengekstrak isi paket **Debian** menggunakan perkakas ini, kita dapat menggunakan beberapa macam perintah. Perintah-perintah tersebut adalah **`-x`** atau **`--extract`**, **`-X`** atau **`--vextract`**, **`-e`** atau **`--control`**, **`-R`** atau **`--raw-extract`**, **`--ctrl-tarfile`** dan **`--fsys-tarfile`**. Namun, untuk pembahasan dalam buku ini tidak akan mencakup keseluruhan dari perintah-perintah tersebut secara rinci.

Untuk mengekstrak keseluruhan isi dari berkas paket meliputi berkas-berkas **control** dan pohon sistem berkas (susunan direktori dan berkas yang terkandung di dalam paket), maka kita dapat menggunakan perintah **`-R`** atau **`--raw-extract`**. Namun, jika kita hanya ingin mengekstrak pohon sistem berkasnya saja, maka perintah yang dapat kita gunakan adalah **`-x`** atau **`--extract`**, **`-X`** atau **`--vextract`** dan **`--fsys-tarfile`**. Sedangkan jika kita hanya ingin mengekstrak berkas-berkas **control** paket saja, maka perintah yang kita gunakan adalah **`-e`** atau **`--control`** dan **`--ctrl-tarfile`**.

Sebagai contoh, ketika kita ingin keseluruhan isi dari berkas paket **chmsee_1.3.0-2_amd64.deb**, maka kita dapat menggunakan perintah sebagaimana berikut:

```
$ dpkg-deb -R chmsee_1.3.0-2_amd64.deb chmsee/
```

Namun, jika kita hanya ingin mengekstrak pohon sistem berkasnya saja, maka kita dapat menggunakan perintah sebagaimana berikut:

```
$ dpkg-deb -x chmsee_1.3.0-2_amd64.deb chmsee/
```

Sedangkan jika kita hanya ingin mengekstrak berkas-berkas **control** dari berkas paket saja, maka perintah yang kita gunakan adalah sebagai berikut:

```
$ dpkg-deb -e chmsee_1.3.0-2_amd64.deb chmsee/DEBIAN/
```

```
Terminal - idnux@debian: ~/Downloads
File  Edit  View  Terminal  Tabs  Help
idnux@debian:~/Downloads$ dpkg-deb --verbose -R chmsee_1.3.0-2_amd64.deb chmsee
./
./usr/
./usr/bin/
./usr/bin/chmsee
./usr/share/
./usr/share/applications/
./usr/share/applications/chmsee.desktop
./usr/share/chmsee/
./usr/share/chmsee/about-dialog.ui
./usr/share/chmsee/book-closed.png
./usr/share/chmsee/book-open.png
./usr/share/chmsee/chmsee-icon.png
./usr/share/chmsee/default-prefs.js
./usr/share/chmsee/helpdoc.png
./usr/share/chmsee/hide-pane.png
./usr/share/chmsee/openfile-dialog.ui
./usr/share/chmsee/setup-window.ui
./usr/share/chmsee/show-pane.png
./usr/share/doc/
./usr/share/doc/chmsee/
./usr/share/doc/chmsee/AUTHORS
./usr/share/doc/chmsee/NEWS.gz
./usr/share/doc/chmsee/README
```

Gambar 4.15: Mengekstrak berkas paket Debian menggunakan dpkg-deb

4.4.1.3. Membangun Paket Debian

Selain beberapa fungsi atau kegunaan yang telah dibahas sebelumnya, ada satu lagi kegunaan dari perkakas **dpkg-deb** ini yang mungkin bisa kita anggap sebagai kegunaan yang paling penting. Kegunaan tersebut adalah untuk membangun paket biner **Debian** atau yang lebih dikenal dengan sebutan paket **Debian** saja. Namun yang perlu untuk diketahui, perkakas **dpkg-deb** ini tidak dapat membangun paket biner langsung dari paket sumber atau berkas kode sumber.

Untuk membangun paket **Debian** ini, ada beberapa hal yang perlu kita lakukan. Hal-hal tersebut adalah:

a. Membuat direktori yang nantinya akan digunakan sebagai tempat kerja dan tempat penyimpanan semua berkas yang akan kita paketkan sebagai paket **Debian**. Direktori ini dapat kita beri nama dengan nama paket atau nama program yang akan kita paketkan dan disarankan agar menggunakan nama mengikuti format nama berkas paket biner **Debian**.
b. Membuat pohon direktori di dalam direktori kerja yang telah kita buat sebelumnya yang berisi semua berkas dan direktori yang kita inginkan terdapat dalam bagian sistem berkas data paket. Penempatan berkas-berkas tersebut sebaiknya tetap mengikuti Standar Susunan Berkas atau *File Hierarchy Standard* dan berkas-berkasnya mengikuti ketentuan yang terdapat dalam kebijakan **Debian** atau ***Debian** Policy*.
c. Mengatur kepemilikan dan perizinan dari setiap berkas dan direktori agar sesuai dengan yang kita inginkan dan juga sesuai dengan ketentuan yang berlaku di dalam sistem ketika berkas-berkas dan direktori-direktori tersebut nantinya terpasang.

d. Membuat direktori dengan nama **DEBIAN** yang nantinya akan kita isi dengan berkas-berkas **control** atau *meta-data*. Di dalam direktori tersebut sekurang-kurangnya harus berisi dengan berkas **control**.
e. Setelah semua berkas yang dibutuhkan siap, kita dapat memulai membangun paket menggunakan perkakas **dpkg-deb**. Untuk membangun paket kita dapat menggunakan perintah **`-b`** atau **`--build`**. Pola perintah yang digunakan adalah sebagai berikut:

```
$  dpkg-deb -b <direktori> [<berkas paket>|<direktori>]
```

   Agar kepemilikan dan perizinan dari berkas-berkas yang merupakan isi dari paket dapat tepat atau paling tidak bisa mendekati sebagaimana ketentuan, maka kita dapat menggunakan perkakas bantu **fakeroot**.

Sebagai contoh, di sini kita akan mencoba untuk membuat paket **hunspell-id** yang merupakan data kamus bahasa Indonesia untuk perkakas pemeriksa ejaan dan penganalisa morfologi **hunspell**. Berkas-berkas utama yang akan kita masukkan dalam paket ini ada dua buah, yaitu berkas **id_ID.aff** yang merupakan berkas yang berisi aturan imbuhan untuk kamus bahasa Indonesia dan berkas **id_ID.dic** yang merupakan berkas yang berisi kamus atau daftar kata bahasa Indonesian. Berkas-berkas ini nanti akan kita tempatkan dalam direktori **/usr/share/hunspell/**. Di dalam paket ini nanti tidak akan kita sertakan berkas-berkas pelengkap seperti berkas dokumentasi, petunjuk (**README**) maupun informasi lisensi. Paket akan menggunakan nomor versi 2.0 dan nomor revisi 2. Arsitektur paket tersebut adalah **all** atau arsitektur independen. Berkas-berkas **control** yang akan kita buat hanya berupa berkas **control** saja. Untuk membuat paket tersebut kita akan melakukan beberapa langkah berikut:

```
$  mkdir hunspell-id_2.0-2_all
$  cd hunspell-id_2.0-2_all
$  mkdir -p usr/share/hunspell
$  mkdir DEBIAN
```

Setelah kita membuat direktori-direktori yang kita perlukan, maka kita dapat menyalin berkas-berkas kamus tersebut ke dalam direktori **/usr/share/hunspell** yang telah kita persiapkan tersebut. Kemudian kita buat juga berkas **control** dan menyimpannya ke direktori **DEBIAN**. Berikut adalah contoh isi berkas **control** yang dapat kita buat:

```
Package: hunspell-id
Version: 2.0-3
Architecture: all
Maintainer: Ali Ahmadi <idnux09@gmail.com>
Installed-Size: 331
Depends: dictionaries-common (>= 0.10)
Suggests: hunspell, openoffice.org-hunspell | openoffice.org-core (>= 2.0.2)
Conflicts: libxul0d (= 1.8.0.11-3), mozilla-browser (<< 1.8+1.1.1-2),
 myspell-id, openoffice.org (<= 1.0.3-2), openoffice.org-core (<< 2.1~m190-1)
Replaces: myspell-id, openoffice.org-spellcheck-id
Provides: hunspell-dictionary, hunspell-dictionary-id, hunspell-dictionary-id
Section: text
Priority: optional
Description: Indonesian dictionary for hunspell
 This is the Indonesian dictionary for use with the hunspell spellchecker
```

Gambar 4.16: Contoh berkas control untuk paket hunspell-id

Setelah semua berkas yang diperlukan telah siap dan diletakkan dalam tempat yang tepat, maka kita dapat memulai membangun paket dengan menggunakan perintah sebagai berikut:

```
$  fakeroot dpkg-deb -b hunspell-id_2.0-2_all/
```

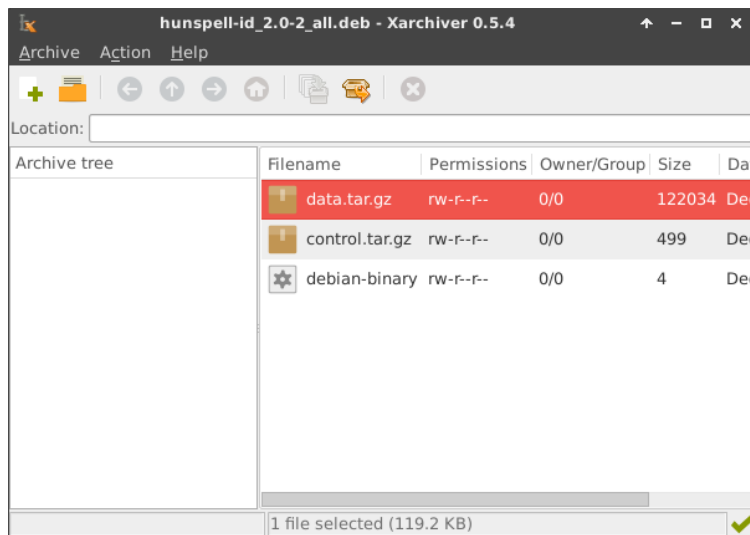
Namun, jika direktori yang kita buat sebelumnya tidak menggungakan mana direktori yang mengikuti format penomoran berkas paket **Debian**, sebagai contoh nama direktorinya adalah **hunspell-id**, maka perintah yang akan kita gunakan adalah sebagai berikut:

```
$  fakeroot dpkg-deb -b hunspell-id/ hunspell-id_2.0-2_all.deb
```

Setelah berkas paket telah berhasil dibuat, maka kita dapat langsung memasang berkas paket tersebut ke dalam sistem dengan perintah sebagaimana berikut:

```
# dpkg -i hunspell-id_2.0-2_all.deb
```

Dalam pembangunan paket ini, kita juga dapat menambahkan beberapa opsi pada perintah **dpkg-deb** yang kita jalankan untuk mengatur proses pembangunan paket dengan menambahkan beberapa ketentuan secara khusus yang akan diterapkan pada berkas paket yang dihasilkan. Opsi-opsi yang bisa kita tambahkan tersebut adalah **`-z`**, **`-S`**, **`-Z`**, **`--uniform-compression`**, **`--deb-format`** dan **`--no-check`**. Sebagai contoh, format kompresi baku yang digunakan untuk paket **Debian** saat ini adalah **xz**. Ketika kita ingin mengubah format kompresi yang akan diterapkan pada paket menjadi **gzip** dan juga tingkat kompresinya menjadi delapan (tingkat kompresi baku untuk format **gzip** adalah sembilan), maka perintah yang kita gunakan adalah sebagai berikut:

```
$ fakeroot dpkg-deb -Zgzip -z8 -b hunspell-id_2.0-2_all/
```

Gambar 4.17: Contoh isi paket Debian dengan format kompresi gzip

DAFTAR PUSTAKA

Hertzog, Raphaël dan Roland Mas. 2015. *The Debian Administrator's Handbook*. Sorbiers:Freexian.

Foster-Johnson, Eric. 2005. *RPM Guide*, (Daring), (http://rpm5.org/docs/rpm-guide.html, diakses tanggal 11 Mei 2016).

Rodin, Josip dan Osamu Aoki. 2015. *Debian New Maintainer's Guide*, (Daring), (https://www.debian.org/doc/manuals/maint-guide/index.en.html, diakses tanggal 28 April 2016).

Jackson, Ian dan Christian Schwarz. 1998. *Debian Policy Manual*, (Daring), (https://www.debian.org/doc/debian-policy/index.html, diakses tanggal 9 Mei 2016).

Hertzog, Raphaël. 2016. *Mastering Debian and Ubuntu*, (Daring), (https://raphaelhertzog.com/mastering-debian/, diakses tanggal 29 Februari 2016).

Beekmans, Gerard dan Bruce Dubbs (Ed). 2016. *Linux From Scratch: Package Management*, (Daring), (http://www.linuxfromscratch.org/lfs/view/development/chapter06/pkgmgt.html, diakses tanggal 19 Februari 2016).

Uekawa, Junichi. 2006. *Debian Library Packaging guide*, (Daring), (https://www.netfort.gr.jp/~dancer/column/libpkg-guide/libpkg-guide.html, diakses tanggal 29 Februari 2016).

Byfield, Bruce. 2006. *Housekeeping utilities for Debian packages*, (Daring), (https://www.linux.com/news/housekeeping-utilities-debian-packages, diakses tanggal 29 Februari 2016).

Srivastava, Manoj. 2009. *Maintainer scripts*, (Daring), (https://people.debian.org/~srivasta/MaintainerScripts.html, diakses tanggal 9 Mei 2016).

Brokmeier, J.Z. 2011. *Stories of Linux: Interview with Ian Murdock on Debian's Early Days*, (Daring), (https://www.linux.com/news/stories-linux-interview-ian-murdock-debians-early-days, diakses tanggal 29 Februari 2016).

Murdock, Ian. 2007. *How package management changed everything*, (Daring), (http://ianmurdock.com/solaris/how-package-management-changed-everything/, diakses tanggal 29 Februari 2016).

Brady, Scot. 2016. *SyrLUG-Contributions-Debian Package Management*, (Daring), (http://www.syrlug.org/contrib/debian-package.html, diakses tanggal 29 Februari 2016).

Pop, Frans. 2010. *Debian Installer internals*, (Daring), (https://d-i.alioth.debian.org/doc/internals/index.html, diakses tanggal 30 Agustus 2016).

Oxer, Jonathan. 2006. *Anatomy of a Debian Package*, [pdf], (Daring), (http://jon.oxer.com.au/sb/modules/talks/attachments/30/20060721-Google-DebianPackages.pdf, diakses tanggal 18 Mei 2016).

Lee, Chr. Clemens. 2005. *Debian Binary Package Building HOWTO*, (Daring), (http://tldp.org/HOWTO/html_single/Debian-Binary-Package-Building-HOWTO/, diakses tanggal 18 Mei 2016).

Akbar, Ade Malsasa. 2016. *Linuxku.com: Seri Manajemen Paket Ubuntu 3: Mengenal Dpkg*, (Daring), (http://www.linuxku.com/2016/05/seri-manajemen-paket-ubuntu-3-mengenal-dpkg.html, diakses tanggal 30 Agustus 2016).

Akkerman, Wichert. 2001. *evolution of the Debian package management system*, (Daring), (http://www.wiggy.net/presentations/2001/DpkgEvolution/html/mgp00005.html, diakses tanggal 18 Mei 2016 lewat http://archive.org/web).

_____. *Package Manager*, (Daring), (https://en.wikipedia.org/wiki/Package_manager, diakses tanggal 19 Februari 2016).

_____. *Package management*, (Daring), (https://en.opensuse.org/Package_management, diakses tanggal 19 Februari 2016).

_____. *Package management system*, (Daring), (https://fedoraproject.org/wiki/Package_management_system, diakses tanggal 19 Februari 2016).

_____. *The Debian GNU/Linux FAQ*, (Daring), (https://www.debian.org/doc/manuals/debian-faq/index.en.html, diakses tanggal 29 Februari 2016).

_____. *DebianPackageManagement*, (Daring), (https://wiki.debian.org/DebianPackageManagement, diakses tanggal 19 Februari 2016).

_____. *A Brief History of Debian*, (Daring), (https://www.debian.org/doc/manuals/project-history/index.en.html, diakses tanggal 29 Februari 2016).

_____. 2013. *15 Practical Examples of “dpkg commands” for Debian Based Distros*, (Daring), (http://www.tecmint.com/dpkg-command-examples/, diakses tanggal 30 Agustus 2016).

_____. *MaintainerScripts*, (Daring), (https://wiki.debian.org/MaintainersScripts, diakses tanggal 9 Mei 2016).

_____. *Linux Package Management*, (Daring), (https://www.linode.com/docs/tools-reference/linux-package-management, diakses tanggal 19 Februari 2016).

Lisensi Publik Creative Commons Atribusi-Berbagi Serupa 4.0 Internasional

Creative Commons Corporation (“**Creative Commons**”) bukanlah suatu firma hukum dan tidak memberikan layanan hukum. Distribusi lisensi publik **Creative Commons** tidak mengisyaratkan hubungan layaknya pengacara-klien maupun jenis hubungan lainnya. **Creative Commons** menyediakan lisensi dan informasi terkait berdasarkan ketentuan “apa adanya (*as-is*)”. **Creative Commons** tidak memberikan jaminan atas lisensi ini, setiap materi berlisensi di bawah syarat dan ketentuan lisensi, atau setiap informasi yang terkait. **Creative Commons** menolak semua tanggung jawab atas kerugian yang muncul karena pengguna-annya.

Lisensi
Dengan menggunakan **Hak Lisensi** (penjelasan di bawah), Anda menerima dan setuju untuk terikat dengan syarat dan ketentuan dari **Lisensi Publik Creative Commons Atribusi-Berbagi Serupa 4.0 Internasional** (“**Lisensi Publik**”). Sejauh **Lisensi Publik** ini dapat dipandang sebagai kontrak, Anda menerima **Hak Lisensi** dengan mempertimbangkan persetujuan Anda atas syarat dan ketentuan berikut, dan **Pemberi Lisensi** memberikan Anda hak-hak tersebut dengan mempertimbangkan manfaat yang diterima **Pemberi Lisensi** dengan menyebarluaskan **Materi Berlisensi** di bawah syarat dan ketentuan berikut.

Bagian 1 – Definisi.

f. **Materi Adaptasi** adalah materi yang memiliki **Hak Cipta** dan Hak-hak Serupa yang diambil dari atau dibuat berdasarkan **Materi Berlisensi** dan **Materi Berlisensi** harus diterjemahkan, diubah, disusun ulang, dialihwujudkan, atau dimodifikasi dengan cara lainnya dengan izin **Pemberi Lisensi** atas **Hak Cipta** dan Hak-hak Serupa. Untuk tujuan **Lisensi Publik** ini, dalam hal **Materi Berlisensi** adalah ciptaan musikal, pertunjukan, atau rekaman suara, **Materi Adaptasi** juga tercipta saat **Materi Berlisensi** dipadukan dengan gambar bergerak.
g. **Lisensi Pengadaptasi** adalah lisensi yang Anda berikan pada **Hak Cipta** dan Hak-hak Serupa Anda pada kontribusi Anda dalam **Materi Adaptasi** sesuai dengan syarat dan ketentuan dalam **Lisensi Publik** ini.
h. **Lisensi Kompatibel BY-SA** adalah salah satu dari lisensi yang dicantumkan pada creativecommons.org/compatiblelicenses, diakui oleh **Creative Commons** sebagai pada pokoknya setara dengan ini.
i. **Hak Cipta dan Hak-hak Serupa** adalah hak cipta dan/atau Hak-hak Serupa yang berkaitan dengan hak cipta termasuk, tanpa terbatas pada, hak untuk mempertunjukan, menyiarkan, merekam suara, dan **Hak Basis Data Sui Generis**, tanpa mengurangi tanda atau kategori hak-hak tersebut. Untuk tujuan **Lisensi Publik** ini, hak-hak yang disebutkan dalam Bagian **2(b)(1)-(2)** tidak termasuk **Hak Cipta** dan Hak-hak Serupa.
j. **Sarana Kontrol Teknologi** adalah tindakan yang, dalam hal tidak adanya otoritas yang berkewajiban, tidak boleh dirusak di bawah ketentuan peraturan perundang-undangan yang memenuhi kewajiban **Pasal 11 Perjanjian**

Hak Cipta WIPO yang berlaku sejak 20 Desember 1996 dan/atau perjanjian internasional yang serupa.

k. **Pengecualian dan Pembatasan** adalah penggunaan yang wajar dan/atau pengecualian atau pembatasan lain atas **Hak Cipta** dan Hak-hak Serupa yang berlaku pada penggunaan Anda atas **Materi Berlisensi**.

l. **Elemen Lisensi** adalah atribut lisensi yang dicantumkan di bawah nama **Lisensi Publik Creative Commons**. Elemen Lisensi dari **Lisensi Publik** ini adalah **Atribusi** dan **Berbagi Serupa**.

m. **Materi Berlisensi** adalah ciptaan di bidang seni dan sastra, basis data, atau materi lain yang menggunakan **Lisensi Publik** ini.

n. **Hak Lisensi** adalah hak-hak yang diberikan kepada Anda berdasarkan syarat dan ketentuan pada **Lisensi Publik** ini, yang terbatas pada seluruh **Hak Cipta** dan Hak-hak Serupa yang berlaku pada penggunaan Anda atas **Materi Berlisensi** dan terbatas pada bagian yang dapat dilisensikan oleh **Pemberi Lisensi**.

o. **Pemberi Lisensi** adalah setiap individu atau entitas yang memberikan hak-hak di bawah **Lisensi Publik**.

p. **Membagikan** adalah menyediakan materi kepada publik dengan cara atau proses apapun yang membutuhkan izin di bawah **Hak Lisensi**, seperti memperbanyak, memamerkan, mempertunjukan, menyebarluaskan, mengomunikasikan, atau mengimpor, dan untuk menyediakan materi kepada publik termasuk melalui cara-cara yang memungkinkan publik untuk mengakses materi dari tempat dan waktu yang mereka pilih sendiri.

q. **Hak Basis Data Sui Generis** adalah hak selain hak cipta yang muncul dari **Direktif Parlemen** dan **Dewan Uni Eropa 11 Maret 1996 No. 96/9/EC** tentang tentang perlindungan hukum atas basis data, sebagaimana diamandemen dan/atau digantikan, juga hak-hak setara lainnya yang berlaku.

r. **Anda** adalah setiap individu atau entitas hukum yang menggunakan **Hak Lisensi** di bawah **Lisensi Publik** ini.

Bagian 2 – Ruang Lingkup.

a. **Ruang lingkup lisensi**.
   1. Sesuai dengan syarat dan ketentuan di dalam **Lisensi Publik** ini, **Pemberi Lisensi** memberikan Anda lisensi yang berlaku di seluruh dunia, bebas royalti, tidak dapat dilisensikan kembali, non-eksklusif, dan tidak dapat dicabut untuk menggunakan **Hak Lisensi** pada **Materi Berlisensi** untuk:
      A. memperbanyak dan Membagikan **Materi Berlisensi**, dalam bentuk utuh maupun sebagian; dan
      B. menciptakan, memperbanyak, dan membagikan **Materi Adaptasi**.
   2. Pengecualian dan Pembatasan. Untuk menghindari keraguan, saat Pengecualian dan Pembatasan berlaku atas penggunaan Anda, **Lisensi Publik** ini tidak berlaku, dan Anda tidak wajib melaksanakan syarat dan ketentuan di dalamnya.
   3. Jangka waktu. Jangka waktu **Lisensi Publik** ini dijelaskan pada Bagian **6(a)**.
   4. Media dan format; modifikasi teknis diperbolehkan. **Pemberi Lisensi** mengizinkan Anda untuk menggunakan **Hak Lisensi** pada seluruh media dan format baik yang telah diketahui saat ini maupun yang akan diciptakan, dan untuk melakukan modifikasi teknis yang diperlukan. **Pemberi Lisensi** melepaskan dan/atau setuju untuk tidak menggunakan hak

atau kewenangan apapun untuk melarang Anda melakukan modifikasi teknis yang diperlukan untuk menggunakan **Hak Lisensi**, termasuk modifikasi teknis yang diperlukan untuk menghilangkan Sarana Kontrol Teknologi. Untuk tujuan **Lisensi Publik** ini, pembuatan modifikasi yang diizinkan Bagian **2(a)(4)** ini tidak berarti menghasilkan **Materi Adaptasi**.

5. Pengguna lain.
    A. Tawaran dari **Pemberi Lisensi – Materi Berlisensi**. Setiap pengguna **Materi Berlisensi** secara otomatis menerima tawaran dari **Pemberi Lisensi** untuk menggunakan **Hak Lisensi** di bawah syarat dan ketentuan dalam **Lisensi Publik** ini.
    B. Tawaran Tambahan dari **Pemberi Lisensi – Materi Adaptasi**. Setiap pengguna **Materi Adaptasi** yang Anda ciptakan secara otomatis menerima tawaran dari **Pemberi Lisensi** untuk menggunakan **Hak Lisensi** pada **Materi Adaptasi** di bawah syarat dan ketentuan dalam **Lisensi Pengadaptasi** yang Anda gunakan.
    C. Tidak ada halangan tambahan. Anda tidak boleh menawarkan atau memberikan syarat dan ketentuan tambahan pada, atau menggunakan Sarana Kontrol Teknologi pada, **Materi Berlisensi** apabila tindakan tersebut menghalangi penggunaan **Hak Lisensi** oleh setiap pengguna lain dari **Materi Berlisensi**.
6. Tanpa dukungan. Tidak satupun hal di bawah **Lisensi Publik** ini memunculkan atau dapat memunculkan izin untuk menyatakan secara langsung maupun tidak langsung bahwa Anda, atau penggunaan Anda atas **Hak Lisensi**, adalah terkait dengan, atau didukung oleh, atau secara resmi diberikan oleh, **Pemberi Lisensi** atau pihak lain yang harus menerima atribusi sebagaimana tercantum dalam Bagian **3(a)(1)(A)(i)**.

b. **Hak lainnya**.
    1. Hak moral, seperti hak atas integritas, tidak dilisensikan di bawah **Lisensi Publik** ini, juga hak atas potret, kerahasiaan, dan/atau hak personal serupa lainnya; walau demikian, sejauh yang dimungkinkan, **Pemberi Lisensi** melepaskan dan/atau setuju untuk tidak menggunakan hak lainnya yang dipegang oleh **Pemberi Lisensi** hingga batasan tertentu untuk mengizinkan Anda menggunakan **Hak Lisensi**, dan bukan sebaliknya.
    2. Hak paten dan merek dagang tidak dilisensikan di bawah **Lisensi Publik**.
    3. Sejauh yang dimungkinkan, **Pemberi Lisensi** melepaskan hak untuk mendapatkan royalti dari Anda dengan menggunakan **Hak Lisensi**, baik secara langsung maupun melalui lembaga manajemen kolektif di bawah ketentuan skema lisensi sukarela atau yang dapat dilepaskan maupun skema lisensi wajib. Dalam kondisi lainnya **Pemberi Lisensi** memiliki hak untuk mendapatkan royalti tersebut.

Bagian 3 – Ketentuan Lisensi.

Penggunaan Anda atas **Hak Lisensi** bergantung pada ketentuan di bawah ini.

a. **Atribusi**.
    1. Apabila Anda Membagikan **Materi Berlisensi** (termasuk yang telah dimodifikasi), Anda wajib:
        A. mencantumkan hal-hal di bawah ini apabila dinyatakan oleh **Pemberi Lisensi** di dalam **Materi Berlisensi**:
            i. identitas para pencipta **Materi Berlisensi** dan pihak lain yang harus menerima atribusi, dengan cara yang sesuai dengan per-

mintaan **Pemberi Lisensi** (termasuk penggunaan nama samaran jika ada);
ii. pemberitahuan hak cipta;
iii. pemberitahuan mengenai **Lisensi Publik** ini;
iv. pemberitahuan mengenai pernyataan jaminan;
v. PSS (pengidentifikasi sumber seragam) atau tautan kepada **Materi Berlisensi** jika memungkinkan;

B. menunjukkan bahwa Anda telah memodifikasi **Materi Berlisensi** dan menunjukkan setiap modifikasi yang sebelumnya pernah dibuat; dan
C. menunjukkan bahwa **Materi Berlisensi** dilisensikan di bawah **Lisensi Publik** ini, termasuk teks, atau PSS dan tautan kepada, dari **Lisensi Publik** ini.

2. Anda dapat melaksanakan ketentuan pada Bagian **3(a)(1)** dengan cara yang sesuai dengan perantara, cara, dan konteks penggunaan Anda dalam Membagikan **Materi Berlisensi**. Sebagai contoh, Anda dapat melaksanakan ketentuan ini dengan memberikan PSS atau tautan kepada sumber yang menyediakan informasi yang dibutuhkan.
3. Apabila dimintakan demikian oleh **Pemberi Lisensi**, Anda harus menghilangkan setiap informasi yang dinyatakan dalam Bagian **3(a)(1)(A)** jika memungkinkan.

b. **BerbagiSerupa**.
Sebagai tambahan dari ketentuan dalam Bagian **3(a)**, apabila Anda Membagikan **Materi Adaptasi** yang Anda ciptakan, ketentuan di bawah ini juga berlaku.
1. **Lisensi Pengadaptasi** yang Anda gunakan harus berupa lisensi **Creative Commons** dengan Elemen Lisensi yang sama, versi ini maupun versi terdahulu, atau **Lisensi Kompatibel BY-SA**.
2. Anda wajib mencantumkan teks, atau PSS atau tautan kepada, **Lisensi Pengadaptasi** yang Anda gunakan. Anda dapat melaksanakan ketentuan ini dengan cara yang sesuai dengan medium, bentuk, dan konteks penggunaan Anda dalam Membagikan **Materi Adaptasi**.
3. Anda tidak boleh menawarkan atau memberikan syarat dan ketentuan tambahan pada, atau menggunakan Sarana Kontrol Teknologi pada, **Materi Adaptasi** yang dapat menghalangi pengguna **Materi Adaptasi** untuk melaksanakan **Lisensi Publik** ini penggunaan hak yang diberikan oleh **Lisensi Pengadaptasi** yang Anda gunakan.

Bagian 4 – Hak Basis Data Sui Generis.

Apabila **Hak Lisensi** mencakup **Hak Basis Data Sui Generis** yang berlaku untuk penggunaan Anda atas **Materi Berlisensi**:

a. untuk menghindari keraguan, Bagian **2(a)(1)** memberikan hak kepada Anda untuk mengambil, menggunakan kembali, memperbanyak, dan Membagikan seluruh atau sebagian besar dari isi basis data;
b. apabila Anda mencantumkan seluruh atau sebagian besar dari isi basis data di dalam basis data dengan **Hak Basis Data Sui Generis** yang dapat Anda gunakan, maka basis data dengan **Hak Basis Data Sui Generis** yang dapat Anda gunakan (namun bukan isinya secara terpisah) adalah **Materi Adaptasi**, termasuk untuk kepentingan Bagian **3(b)**; dan
c. Anda harus melaksanakan ketentuan dalam Bagian **3(a)** apabila Anda Membagikan seluruh atau sebagian besar dari isi basis data.

Untuk menghindari keraguan, Bagian **4** ini menambahkan dan tidak menggantikan kewajiban Anda di bawah **Lisensi Publik** di mana **Hak Lisensi** termasuk **Hak Cipta** dan Hak-hak Serupa lainnya.

Bagian 5 – Pernyataan Jaminan dan Batasan Tanggung Jawab.

a. **Kecuali dilaksanakan dengan cara lain secara terpisah oleh Pemberi Lisensi, sejauh yang dimungkinkan, Pemberi Lisensi memberikan Materi Berlisensi "apa adanya" dan "sebagaimana tersedia", dan tidak memberikan pernyataan atau jaminan apapun terkait Materi Berlisensi, baik secara tegas, secara tersirat, berdasarkan hukum, atau lainnya. Hal ini termasuk, tanpa terbatas pada, jaminan atas hak, jual beli, kecocokan untuk tujuan tertentu, tidak adanya pelanggaran, tidak adanya pengelabuan atau cacat lainnya, keakuratan, atau adanya atau tidak adanya kesalahan, baik yang diketahui maupun tidak. Dalam hal pernyataan jaminan tidak dapat dilaksanakan secara penuh atau sebagian, pernyataan ini tidak berlaku terhadap Anda.**
b. **Sejauh yang dimungkinkan, Pemberi Lisensi tidak dapat dianggap bertanggung jawab kepada Anda untuk tindakan hukum apapun (termasuk, tanpa terbatas pada, kelalaian) atau untuk setiap kehilangan, kerugian, pengeluaran, atau kerusakan yang terjadi secara langsung, khusus, tidak langsung, sebagai kecelakaan, akibat, hukuman, pembebasan, atau lainnya yang muncul dari Lisensi Publik ini atau penggunaan Materi Berlisensi, walau pemberi lisensi telah memberitahukan kemungkinan terjadinya kehilangan, kerugian, pengeluaran, atau kerusakan tersebut. Dalam hal pembatasan tanggung jawab tidak dapat dilaksanakan secara penuh atau sebagian, batasan ini tidak berlaku terhadap Anda.**
c. Pernyataan jaminan dan batasan tanggung jawab di atas ini dapat diinterpretasikan dengan cara, sejauh yang dimungkinkan, yang paling dekat dengan pernyataan dan pelepasan dari seluruh tanggung jawab sepenuhnya.

Bagian 6 – Jangka Waktu dan Penghentian.

a. **Lisensi Publik** ini berlaku sepanjang jangka waktu **Hak Cipta** dan Hak-hak Serupa yang dilisensikan di bawahnya. Walau demikian, apabila Anda lalai dalam melaksanakan ketentuan **Lisensi Publik** ini, maka hak Anda di bawah **Lisensi Publik** ini akan secara otomatis berakhir.
b. Dalam hal hak Anda untuk menggunakan **Materi Berlisensi** telah berakhir sesuai Bagian **6(a)**, **Lisensi Publik** ini dapat berlaku kembali:
    1. secara otomatis pada tanggal ditanggulanginya kelalaian yang dilakukan, atau dalam waktu 30 hari setelah Anda mengetahui kelalaian yang dilakukan; atau
    2. berdasarkan pemberlakukan kembali oleh **Pemberi Lisensi**.

    Untuk menghindari keraguan, Bagian **6(b)** tidak berpengaruh pada setiap pelanggaran atas hak yang dimiliki oleh **Pemberi Lisensi** yang terjadi akibat kelalaian Anda melaksanakan ketentuan dalam **Lisensi Publik** ini.
c. Untuk menghindari keraguan, **Pemberi Lisensi** dapat menawarkan **Materi Berlisensi** di bawah syarat dan ketentuan yang terpisah atau berhenti menyebarluaskan **Materi Berlisensi** di setiap waktu; namun, hal ini tidak akan menghentikan keberlakuan **Lisensi Publik** ini.
d. Bagian **1**, **5**, **6**, **7**, dan **8** akan terus berlaku setelah **Lisensi Publik** ini berakhir.

Bagian 7 – Syarat dan Ketentuan Lain.

a. **Pemberi Lisensi** tidak terikat di bawah setiap syarat atau ketentuan tambahan atau syarat atau ketentuan yang berbeda yang Anda sarankan kecuali secara tegas disetujui demikian.
b. Setiap aturan, kesepakatan, atau perjanjian terkait **Materi Berlisensi** yang tidak tercantum di sini berlaku secara terpisah dan tidak bergantung pada syarat dan ketentuan di bawah **Lisensi Publik** ini.

Bagian 8 – Interpretasi.

a. Untuk menghindari keraguan, **Lisensi Publik** ini tidak bertujuan untuk, dan tidak dapat dianggap bertujuan untuk, mengurangi, membatasi, menghalangi, atau memaksakan ketentuan dari setiap penggunaan **Materi Berlisensi** yang secara hukum dapat diciptakan tanpa izin di bawah **Lisensi Publik** ini.
b. Sejauh yang dimungkinkan, apabila ketentuan di bawah **Lisensi Publik** ini dinilai tidak dapat dilaksanakan, maka kewajiban pelaksanaan ketentuan harus dilakukan dengan cara yang sesuai. Apabila ketentuan yang ada tidak dapat dilaksanakan sama sekali, maka ketentuan tersebut ditiadakan dari **Lisensi Publik** ini namun tidak berpengaruh pada pelaksanaan syarat dan ketentuan lainnya.
c. Syarat atau ketentuan di bawah **Lisensi Publik** ini tidak dapat dilepaskan dan tidak ada kelalaian yang diizinkan kecuali secara tegas disetujui demikian oleh **Pemberi Lisensi**.
d. **Lisensi Publik** ini tidak memunculkan atau dapat dianggap sebagai pembatasan dari, atau pelepasan atas, setiap hak dan kekebalan yang berlaku untuk **Pemberi Lisensi** atau Anda, termasuk dari proses hukum menurut jurisdiksi atau kewenangan apapun.

Creative Commons bukanlah pihak dalam lisensi publik. Walau demikian, **Creative Commons** dapat memilih untuk menggunakan salah satu dari lisensi publik yang ada untuk materi yang dipublikasikan oleh kami dan dalam hal demikian dianggap sebagai "**Pemberi Lisensi**." Teks dari lisensi publik **Creative Commons** disebarluaskan di bawah Dedikasi Domain Publik CC0. Kecuali untuk tujuan tertentu dalam hal menyatakan bahwa materi tersebut disebarluaskan di bawah lisensi publik **Creative Commons** atau sebagaimana diizinkan oleh aturan **Creative Commons** yang dicantumkan pada creativecommons.org/policies, **Creative Commons** tidak mengizinkan penggunaan merek dagang "**Creative Commons**" atau setiap merek dagang atau logo **Creative Commons** tanpa izin tertulis termasuk, tanpa terbatas pada, dalam hubungannya dengan setiap modifikasi tanpa izin yang dilakukan terhadap lisensi publik atau setiap aturan, kesepakatan, atau perjanjian lainnya terkait penggunaan materi berlisensi. Untuk menghindari keraguan, paragraf ini tidak menjadi bagian dari lisensi publik.

Creative Commons dapat dihubungi melalui creativecommons.org.

TENTANG PENULIS

Nama saya Ali Ahmadi. Saya merupakan pengguna pemula sistem operasi **GNU/Linux**. Saya dilahirkan di Pati, Jawa Tengah pada tahun 1988. Saat ini saya bekerja sebagai Aparatur Sipil Negara (**ASN**) di sebuah kementerian. Pertama kali saya mengenal sistem operasi **GNU/Linux** dari sebuah majalah komputer ternama pada tahun 2004, namun saya baru dapat menggunakannya secara langsung pada sekitar akhir tahun 2006. Distribusi **GNU/Linux** yang pertama kali saya gunakan adalah **Mandriva Linux**. Saat ini, saya menggunakan distribusi **Arch Linux** dan **Debian GNU/Linux**.

Kontak Penulis:
Surel (*Email*) : idnux09@gmail.com
Telegram : @idnux atau https://telegram.me/idnux
Blog Web : https://idnux.wordpress.com